Normandie Université

THÈSE

Pour obtenir le diplôme de doctorat

Spécialité Informatique

Préparée au sein de ED MIIS

Vers des agents cognitifs, affectifs et sociaux dans la simulation

Présentée et soutenue par

Mathieu Bourgais

Thèse soutenue publiquement le 30 Novembre 2018 devant le jury composé de		
M. / Jean-Pierre MULLER,	Directeur de recherche / CIRAD, Montpellier, France	Rapporteur
M. / Michel OCCELLO,	Professeur d'Université / LCIS, Université Grenoble Alpes, France	Rapporteur
M. / Frédéric AMBLARD,	Professeur d'Université / IRIT, Université Toulouse 1, France	Examinateur
Mme. / Magalie OCHS,	Maitre de conférence / LSIS, Polytech Marseille, Aix-Marseille Université	Examinateur
M. / Nicolas SABOURET,	Professeur d'Université / LIMSI, Université Paris-Sud, France	Examinateur
M. / Laurent VERCOUTER,	Professeur d'Université / LITIS, INSA Rouen Normandie, France	Directeur de thèse
M. / Patrick TAILLANDIER,	Chercheur / INRA, Toulouse, France	Co-encadrant de thèse

Thèse dirigée par Laurent Vercouter et Patrick Taillandier

cnrs dépasser les frontières

PROJET FINANCÉ PAR L'ANR · PROJECT FUNDED BY THE ANR · ANR

INSA ROUEN NORMANDIE

MIIS

litis

idees

Résumé

Au cours des dernières années, l'utilisation de simulations à base d'agents pour étudier les systèmes sociaux s'est étendue à de nombreux domaines (géographie, écologie, sociologie, économie, etc.). Ces simulations visent à reproduire des situations réelles impliquant des acteurs humains ; il est donc nécessaire d'y intégrer des agents complexes reproduisant le comportement des personnes simulées. Par conséquent, des notions telles que la cognition, les émotions, la personnalité, les relations sociales ou les normes doivent être prises en compte. Pour autant, il n'existe actuellement aucune architecture d'agent intégrant toutes ces caractéristiques et pouvant être utilisée par la majorité des modélisateurs, y compris ceux n'étant pas expert en programmation informatique.

Dans cette thèse, l'architecture BEN (Behavior with Emotions and Norms) est présentée pour répondre à cette question. Il s'agit d'une architecture modulaire basée sur le modèle BDI de la cognition avec des modules pour ajouter des émotions, de la contagion émotionnelle, une personnalité, des relations sociales et des normes au comportement des agents. Ces dimensions comportementales sont formalisées de manière à ce qu'elles puissent fonctionner ensemble pour produire un comportement crédible dans le contexte des simulations sociales. L'architecture est implémentée dans la plate-forme de simulation GAMA afin de la rendre utilisable par la communauté des simulations sociales. Enfin, BEN est utilisé pour étudier deux cas d'évacuation d'une boîte de nuit en feu, montrant que l'architecture est actuellement utilisable à travers son implémentation dans GAMA et qu'elle permet aux modélisateurs de reproduire des situations réelles impliquant des acteurs humains.

Abstract

Over the last few years, the use of agent-based simulations to study social systems has spread to many domains (e.g. geography, ecology, sociology, economy). These simulations aim to reproduce real life situations involving human beings and thus need to integrate complex agents to match the behavior of the people simulated. Therefore, notions such as cognition, emotions, personality, social relations or norms have to be taken into account, but currently there is no agent architecture that could incorporate all these features and be used by the majority of modelers, including those with low levels of skills in programming.

In this thesis, the BEN (Behavior with Emotions and Norms) architecture is introduced to tackle this issue. It is a modular architecture based on the BDI model of cognition featuring modules for adding emotions, emotional contagion, personality, social relations and norms to agent behavior. These behavioral dimensions are formalised in a way so they may operate together to produce a believable behavior in the context of social simulations. The architecture is implemented into the GAMA simulation platform in order to make it usable by the social simulation community. Finally, BEN is used to study two cases of evacuation of a nightclub on fire, showing it is currently usable throught its implementation into GAMA and it enables modelers to reproduce real life situations involving human actors.

Table des matières

Liste des figures

Liste des tableaux

Introduction

Contexte

L'étude des systèmes complexes implique de les modéliser, c'est à dire de réduire le problème étudié à une représentation abstraite simplifiée [TDZ08]. Une modélisation classique consiste à représenter le système étudié en utilisant les mathématiques ; le système est alors représenté par un ensemble d'équations décrivant son fonctionnement global. La simulation numérique d'un tel système consiste alors à résoudre le système d'équation lié, avec des valeurs initiales particulières.

La simulation numérique de modèles mathématiques représente un outil scientifique permettant de tester des hypothèses sur un environnement virtuel contrôlé. La résolution d'équations par un ordinateur donnera toujours le même résultat pour les mêmes paramètres d'entrée. De plus, ces calculs sont peu coûteux en temps et en argent, permettant de réaliser plusieurs tests simplement afin d'affiner le modèle proposé. Pour autant, la modélisation mathématique d'un problème possède plusieurs limites :

— Il s'agit d'une approche globale d'un problème ; le système est étudié dans un monde "idéal" et modélisé sur un comportement d'ensemble sans prendre en compte les interactions locales.

— Le problème n'est modélisé que sur un seul niveau d'analyse (soit macroscopique, soit microscopique).

— Les éléments hétérogènes d'un système sont difficilement pris en compte.

— Le résultat ne permet pas de phénomènes émergents et ne donne pas d'indications fines sur les interactions locales entre les éléments du système.

Ces dernières années, la simulation à base d'agents est devenue un puissant outil scientifique [BLPM02] utilisé à travers de multiples disciplines. Le principe de ces simulations est de représenter les composants d'un problème sous la forme de systèmes informatiques, situés dans un environnement et capables d'actions autonomes, appelés agents [Woo09]. Ainsi, au lieu de traduire en équations le système étudié, celui-ci est décrit par un ensemble d'agents hétérogènes évoluant de façon autonome dans un environnement partagé. Ce sont alors les interactions entre ces agents qui permettent de reproduire le phénomène étudié.

Cette approche de modélisation a été utilisée dans de nombreuses disciplines : afin de reproduire des phénomènes physiques comme la mécanique des milieux granulaires [Bre02] ou l'hydrodynamique du ruissellement [Ser00], afin d'étudier des systèmes biologiques [KPWvL01] ou encore dans l'étude des systèmes financiers [Hom06]. L'intérêt est de combiner des modélisations "idéales" d'un système, fondées sur des équations mathématiques, avec une modélisation descriptive, centrée sur les interactions à l'intérieur du modèle [TDZ08]. Plutôt que de regarder le résultat de la résolution d'une équation, les simulations à base d'agents permettent d'observer l'évolution d'un système à plusieurs niveaux de complexité en même temps.

La simulation à base d'agent permet donc de combler les limites montrées par la simulation numérique de modèles mathématiques. En particulier, la modélisation et la simulation à base d'agent possèdent plusieurs avantages :

— Des agents hétérogènes en description et en comportement permettent d'étudier des systèmes possédant des éléments de différentes natures interagissant entre eux.

— Les interactions entre les éléments du système sont au centre du modèle.

— Il est possible d'imbriquer des modèles pour obtenir une analyses sur différents niveaux, avec des modèles microscopiques sur lesquels sont fondés des modèles macroscopiques.

— La modélisation descriptive permet de reproduire le plus fidèlement un système, sans se placer dans un monde "idéal".

Simulations sociales

Une utilisation particulière des simulations à base d'agent porte sur l'étude de systèmes impliquant des acteurs humains ; la communauté parle alors de simulations sociales et d'agents sociaux [GT05]. Là encore, les simulations sociales sont utilisées dans de nombreux domaines : dans la criminologie [MHS10], dans la diffusion d'opinions et d'informations à travers les médias sociaux [RHSV15], dans la simulation militaire [CM10] ou encore dans l'agriculture [KBSP14]. Ces différents exemples d'utilisation montrent que la simulation sociale peut concerner des systèmes de différentes tailles, allant de quelques dizaines d'agents à plusieurs milliers.

L'atout majeur des simulations sociales est de permettre aux chercheurs de travailler sur des systèmes complexes impliquant des acteurs humains de façon simple et peu coûteuse. Sans cela, il serait nécessaire d'organiser des reproductions des phénomènes étudiés ce qui pose deux problèmes : soit les personnes prises pour la reproduction sont mises au courant de leur statut d'acteur et les résultats obtenus seront faussés, soit ces personnes ne sont pas mises au courant de la situation ce qui n'est pas éthique. La simulation sociale permet alors de reproduire les situations étudiées, sans avoir besoin de vrais acteurs et en obtenant des résultats plus rapidement qu'en réalisant des reproductions. Se pose alors la problématique de simuler de façon crédible le comportement d'acteurs humains afin d'obtenir des résultats de simulation les plus proches possibles du système réel, tout en assurant un temps de calcul raisonnable.

Les utilisateurs des simulations sociales sont en grande partie des modélisateurs issus des sciences sociales, qui ne sont donc pas des informaticiens. Pour faciliter l'utilisation de cette approche, des plateformes de modélisation et de simulations ont été développées. Celle-ci offrent un langage de programmation et un environnement permettant de définir des agents puis de les simuler, sans avoir besoin d'implémenter de nouveau les interpréteurs et compilateurs permettant de passer d'un modèle à sa simulation. Les plateformes Repast [Col03], MASON [LCRP+05], Cormas [BBLPB16], NetLogo [WE99] ou encore GAMA [GTG+13] font partie des plus populaires dans la communauté.

Rendre des agents informatiques crédibles dans la simulation d'acteurs humains implique l'intégration de processus complexes simulant le raisonnement humain [Sun06]. Cette vision s'appuie sur le principe EROS (Enhancing the Realism Of Simulation) [Jag17] qui se pose en opposition du principe KISS (Keep it Simple, Stupid) [Axe97], en proposant d'intégrer des notions psychologiques dans la définition d'agents sociaux. Sans cela, les acteurs humains sont considérés comme un ensemble d'éléments simples, par exemple comme des particules évoluant grâce à des phénomènes de champs de forces [GL10].

Les sciences cognitives ont montré qu'une première dimension importante dans la prise de décision et le raisonnement est la cognition [Gar93]. C'est en prenant en compte des concepts comme la mémoire, le langage, le raisonnement, l'apprentissage, l'intelligence la perception ou encore la résolution de problème qu'une personne prend des décisions. Cette cognition a été formalisée et intégrée sous la forme d'architectures et de modèles pour le comportement d'agents informatiques, notamment à travers l'architecture SOAR [LNR87] ou le modèle BDI [Bra87].

Pour autant, les avancées en neurosciences [Dam94] montrent que la prise de décision n'est pas uniquement due à une cognition raisonnée. Des dimensions affectives comme les émotions ou la personnalité ainsi que des dimensions sociales comme les relations envers les autres ou la contagion émotionnelle sont à prendre en compte. Ce sont ces dimensions qui permettent d'expliquer qu'une personne ne prend pas toujours la décision la plus raisonnable dans une situation, même en ayant une perception importante des éléments en jeu.

Certaines de ces dimensions de la prise de décision ont été utilisées par des modélisateurs dans la définition du comportement d'agents sociaux. Le modèle BDI a déjà été utilisé dans différents cas d'application des simulations sociales [AG16], tout comme les émotions [BTVA18], les relations sociales [OSC09b] ou encore la contagion émotionnelle [LLB11]. Seulement, toutes ces dimensions, auxquelles viennent s'ajouter la gestion des normes et la personnalité, n'ont pour le moment pas été combinées dans le développement du comportement d'agents sociaux. Ces travaux sont le plus souvent réalisés pour un cas d'étude précis, donc difficilement généralisables, et n'ont pas été intégrés dans les plateformes de modélisation et de simulation utilisées par la communauté. Il est alors compliqué, pour des modélisateurs non experts en programmation, de réutiliser ces modèles de comportement ou de les combiner pour améliorer les résultats obtenus par la simulation sociale.

Problématiques scientifiques

Dans ce contexte, plusieurs problématiques scientifiques se posent :

Afin de rendre plus crédible le comportement des agents simulant des acteurs humains dans les simulations sociales, il est nécessaire d'intégrer à leur comportement des processus reproduisant les dimensions psychologiques de la prise de décision, en suivant le principe EROS. Cela implique tout d'abord d'avoir une représentation formelles des dimensions cognitives, affectives et sociales que l'agent pourra manipuler dans le but de prendre une décision.

Problématique 1 : Formaliser les notions relatives à la cognition, aux émotions, à la personnalité, à la contagion émotionnelle, aux relations sociales et à la gestion des normes dans un même cadre.

En plus d'une représentation formelle, la prise de décision à travers les dimensions cognitives, affectives et sociales, repose sur des processus qui permettent de manipuler les connaissances de l'agent. Ainsi, ces processus doivent être identifiés et intégrés à une architecture comportementale pour que les agents sociaux reproduisent de façon crédible la prise de décision d'acteurs humains. De plus, cette architecture doit être suffisamment générique pour s'adapter à différents cas d'utilisation. Pour cela, elle doit être paramétrable par les modélisateurs l'utilisant.

Problématique 2 : Intégrer les notions cognitives, affectives et sociales dans une architecture comportementale pour modéliser la prise de décisions d'agents simulant des acteurs humains.

Problématique 3 : Rendre l'architecture comportementale le plus générique possible afin de faciliter son utilisation dans différents domaines.

Enfin, cette architecture étant développée pour la simulation sociale, il est impératif qu'elle soit utilisable par cette communauté qui n'est pas nécessairement informaticienne. Pour cela, l'architecture comportementale doit être intégrée à une plateforme de modélisation et de simulation à base d'agent tout en n'alourdissant pas les temps de calculs de façon excessive.

Problématique 4 : Proposer une implémentation de cette architecture qui soit la plus accessible possible vis à vis d'un public non informaticien et qui puisse être utilisée avec un temps d'exécution raisonnable.

Contributions

Pour résoudre ces problèmes liés à l'intégration de dimensions cognitives, affectives et sociales dans la prise de décision d'agents sociaux, nous proposons dans cette thèse la définition d'une architecture comportementale pour la simulation sociale : BEN (Behavior with Emotions and Norms). La réalisation de BEN implique les éléments suivants :

— Définition d'un formalisme intégrant les concepts cognitifs, affectifs et sociaux pris en compte dans la prise de décision.

— Définition d'une architecture comportementale incluant des notions cognitives, affectives et sociales : BEN (Behavior with Emotions and Norms).

— Implémentation de BEN dans la plateforme de modélisation et de simulation GAMA.

— Utilisation de BEN et de son implémentation pour l'étude d'évacuation de bâtiments.

Plan du manuscrit

Ce manuscrit est organisé de la façon suivante :

Chapitre 1 : Simulations sociales à base d'agents. Le chapitre 1 revient en détails sur les notions liées aux simulations à base d'agents et les simulations sociales. En particulier, les différentes approches proposées dans la littérature afin de simuler des acteurs humains sont présentées et commentées.

Chapitre 2 : Les dimensions de la prise de décision. Le chapitre 2 présente les notions psychologiques, affectives et sociales intervenant dans la prise de décision chez l'Homme. Le chapitre propose une présentation chronologique des avancées dans ce domaine tout en mettant en lumière leur intégration dans des systèmes informatiques.

Chapitre 3 : Formaliser la prise de décision d'un agent social dans une simulation. Le chapitre 3 introduit le formalisme que nous proposons pour représenter les dimensions cognitives, affectives et sociales de la prise de décision. Le chapitre s'attarde en particulier sur la représentation statique des données en jeu.

Chapitre 4 : BEN : une architecture pour des agents au comportement cognitif, affectif et social. Le chapitre 4 présente l'architecture BEN. Cette architecture repose sur le formalisme exposé dans le chapitre 3 en proposant des processus permettant à l'agent de prendre une décision en fonction des évolutions dynamiques de son environnement.

Chapitre 5 : Implémentation de BEN dans GAMA. Le chapitre 5 explique la façon dont l'architecture BEN a été implémentée dans la plateforme de modélisation et de simulation GAMA. En particulier, le chapitre revient sur l'implémentation des différents processus de l'architecture ainsi que sur l'ordre d'exécution de ses composants mais aussi sur les fonctions et procédures ajoutées à la plateforme permettant à un modélisateur d'utiliser l'architecture BEN.

Chapitre 6 : Application de BEN à l'évacuation de bâtiments. Le chapitre 6 expose deux cas d'utilisation de BEN et de son implémentation dans GAMA pour simuler une évacuation de bâtiment. Un modèle d'évacuation est défini dans le détail sur un premier exemple d'évacuation avant d'être adapté sur un second cas d'étude, montrant l'adaptabilité de BEN ainsi que sa capacité à reproduire de façon crédible des situations impliquant des acteurs humains avec un temps de calcul raisonnable.

Enfin, une conclusion ainsi que des perspectives sont évoquées à la fin du manuscrit.

Première partie

État de l'art

Chapitre 1

Simulations sociales à base d'agents

Sommaire

Étudier des situations impliquant des acteurs humains n'est pas une chose simple en pratique. Prenons l'exemple de l'évacuation d'un stade sportif. Pour vérifier que les sorties sont bien placées et en nombre suffisant avant une catastrophe, deux cas de figures se posent : soit un incident est reproduit sans que les personnes testées ne soient prévenues, ce qui n'est pas éthique, soit les personnes testées sont prévenues qu'il s'agit d'un test, ce qui fausse leur réaction.

Un moyen de contourner le problème est de simuler grâce à l'informatique le cas à étudier [AP06]. Se pose alors le problème de la modélisation des acteurs humains, et de leurs comportements dans le cas d'étude. Pour répondre à cette question, la simulation sociale à base d'agents, qui se fonde sur les systèmes multi-agents, une branche de l'intelligence artificielle, est prometteuse.

Cette section présente le principe des simulations à base d'agents, en partant de ses origines dans les systèmes multi-agents jusqu'à son évolution dans les sciences sociales, donnant naissance aux simulations sociales, domaine dans lequel s'inscrit cette thèse. Pour cela, une première partie propose une introduction au domaine scientifique des simulations à base d'agent et une seconde partie s'intéresse aux différents formalismes utilisés pour décrire des comportements dans les travaux sur les agents informatiques.

1.1 Simuler des humains avec des agents informatiques

1.1.1 Les systèmes multi-agents

Les systèmes multi-agents (SMA) représentent un champ scientifique faisant partie de l'intelligence artificielle (IA) qui reposent sur la notion d'agent logiciel. Bien qu'il existe de nombreuses

définitions de la notion d'agent, celle proposée par Wooldridge [Woo09], traduite dans la définition 1, fait consensus :

Définition 1. un **agent** est un système informatique, situé dans un environnement, capable d'actions autonomes pour atteindre ses objectifs.

Cette définition est complétée par Wooldridge [Woo09] et Ferber [Fer97] qui indiquent qu'un agent est une entité informatique qui :

— évolue au sein d'un environnement informatique, noté W, qui peut être ouvert.

— possède une représentation au moins partielle de W.

— peut agir sur W.

— peut percevoir dynamiquement au moins partiellement W.

— peut communiquer avec les autres agents évoluant dans W.

— possède des ressources propres.

— possède ses propres objectifs.

— possède ses propres compétences.

— est capable de mettre en lien ses objectifs, ses ressources et ses compétences ainsi que les informations acquises de W pour réaliser une tâche de façon autonome.

Wooldridge continue sa définition en catégorisant les comportements possibles d'un agent :

— Un comportement **réactif** où l'agent ne fait que répondre aux modifications de l'environnement.

— Un comportement **proactif** où l'agent prend l'initiative pour satisfaire ses objectifs.

— Un comportement **social** où l'agent va interagir avec les autres agents pour atteindre ses objectifs.

Une fois la notion d'agent posée, Wooldridge définit un système multi-agent par la définition suivante 2 :

Définition 2. Un **système multi-agent** est un ensemble d'agents interagissant pour résoudre un problème.

Ainsi, un SMA est un ensemble d'entités hétérogènes, autonomes et partageant un même environnement. Ces agents peuvent avoir localement des objectifs différents, des types et des complexités de comportements inégaux et ne pas avoir les mêmes capacités de perception. Aussi, ce système est ouvert, permettant à un agent de quitter ou de rejoindre le système dynamiquement.

Deux points de vue s'opposent dans la conception d'un SMA : les systèmes "top-down", qui partent du comportement macroscopique attendu pour définir les interactions entre les agents, et les systèmes "bottom-up", qui partent de la définition du comportement de chaque agent et de leurs interactions, au niveau microscopique, pour obtenir un comportement macroscopique qui résout le problème étudié. Dans le premier cas, c'est le système qui indique aux agents comment interagir alors que dans le second cas, ce sont les interactions entre les agents qui vont déterminer la direction du système. En fonction du point de vue, un agent va être plus ou moins autonome, coopératif et avec plus ou moins de complexité dans son comportement.

Enfin, il convient de noter qu'il existe de nombreuses plateformes informatiques dédiées au développement de SMA [KB15] comme JACK [HRHL01], JADE [BBCP05] ou Jadex [PBL05] parmi les plus populaires. Chaque plateforme possède ses spécificités, que ce soit sur la complexité du comportement qu'il est possible de définir où sur le nombre d'agents gérés.

1.1.2 La simulation à base d'agents

La modélisation d'un problème complexe est l'une des deux composantes majeures de la démarche scientifique, avec l'expérimentation [TDZ08]. Modéliser un système consiste à en proposer une version abstraite, simplifiée, et permettant de répondre à une question particulière sur ce système [P+72]. Trois types de modèles sont a distinguer :

— Les modèles normatifs : le système est représenté par un ensemble de règles visant à abstraire le problème dans un monde "idéal".

— Les modèles descriptifs : les entités du système sont décrites telle qu'elles sont. Le modèle est donc plus proche du système à abstraire mais moins facile à analyser.

— Les modèles explicatifs : le modèle est créé avec pour but d'expliquer le phénomène étudié dans les termes du système modélisé.

Ainsi, il n'existe pas de modèle absolu pour un système donné. Le modèle dépend de la question posée sur ce système complexe, de la théorie sur laquelle s'appuyer ainsi que sur les données disponibles sur ce système [CHSN97].

L'apport des SMA permet de proposer des modèles à base d'agents, c'est à dire, comme le définit Treuil [TDZ08], utiliser des SMA pour modéliser le problème étudié. Une fois le problème modélisé, il convient de le simuler pour le mettre en pratique dans un environnement contrôlé. Une simulation a plusieurs objectifs [Sim06] :

— Valider, vérifier ou évaluer une hypothèse, sous-tendue par une théorie, sur un modèle d'un système complexe.

— Communiquer et visualiser un résultat en mettant "en mouvement" un modèle.

— Comprendre, expliquer et explorer le fonctionnement d'un système complexe dont la simulation est une reproduction plus facilement manipulable.

— Contrôler et piloter des évolutions possibles du système de référence, dans le cadre d'un support à une prise de décision.

— Prévoir, prédire et anticiper les évolutions possibles d'un système, même dans des cas particuliers extrêmes.

Dans le cadre de modèles à base d'agent, leur mise en pratique se fait via des simulations à base d'agent qui sont définis, d'après Treuil [TDZ08], selon la définition :

Définition 3. Une **simulation à base d'agent** est l'activité au cours de laquelle, selon un protocole et un objectif précis, une plateforme informatique exécute un modèle à base d'agents afin d'en recueillir les sorties pour un jeu d'entrées donné.

Par définition, ces simulations à base d'agents sont des systèmes "bottom-up" fondés sur des modèles descriptifs puisqu'ils visent à reproduire des situations réelles. Ces arguments ont conduit la communauté des sciences sociales à s'intéresser aux simulations à base d'agents afin d'étudier des problèmes impliquant des acteurs humains, fondant ainsi le domaine des simulations sociales [Dav02] [GT05], qui est définit par la définition 4 :

Définition 4. Une **simulation sociale** est une simulation à base d'agents sociaux, définis via la définition 5.

Définition 5. Un **agent social** est un agent informatique qui modélise un humain.

1.2 Des outils dédiés à la simulation

Les simulations sociales nécessitent des outils informatiques spécifiques permettant de gérer de grandes quantités d'agents dans des temps de calcul raisonnables tout en ayant des comportements complexes. Il est aussi nécessaire d'avoir des outils d'analyse permettant d'exploiter les

résultats obtenus par simulation. Enfin, puisque la communauté des simulations sociales, venant majoritairement des sciences sociales, n'est pas nécessairement experte en programmation, il est intéressant d'avoir des outils simples à prendre en main.

Les plateformes SMA mentionnées précédemment ne sont pas adaptées à la simulation : elles n'ont pas été conçues pour passer à l'échelle, n'ont pas toujours beaucoup d'options de visualisation ou de stockage des résultats et sont tournées vers une communauté d'informaticiens. Pour contourner cela, de nouvelles plateformes ont été proposées spécialement pour la simulation à base d'agents.

Dans cette section, des frameworks spécialisés ainsi que certaines de ces plateformes de simulation à base d'agents parmi les plus populaires sont présentées. Aussi, la question du comportement des agents étant centrale en simulation sociale, la troisième partie de cette section détaille la façon dont ces plateformes traitent ce problème.

1.2.1 Présentation de frameworks pour la simulation

La première approche proposée pour réaliser des simulations à base d'agents a été de créer des frameworks améliorant les langages de programmation déjà existants. Deux de ces frameworks, parmi les plus utilisés par la communauté des simulations à base d'agents, sont présentés dans cette section.

Repast

Repast [Col03][NHCV07] est une plateforme développée en continue depuis 2002 à l'Université de Chicago prenant la forme d'une boite à outil permettant la définition de simulations à base d'agent en Java, Microsoft .NET et python, disponible gratuitement et open-source.

L'utilisateur définit ses agents comme des objets Java, avec un ensemble de caractéristiques et de méthodes. Un fichier XML permet ensuite de décrire l'environnement, l'initialisation ainsi que les liens entre les agents. Enfin, un second fichier XML permet de décrire l'affichage ainsi que le stockage des données en sorties de la simulation. La plateforme s'occupe alors d'exécuter les agents dans l'ordre spécifié.

Repast étant une boite à outil, l'interface graphique est limitée. De plus, c'est une plateforme de bas niveau, qui impose une bonne connaissance de la programmation. Repast est donc une proposition qui ne pose aucune contrainte à ses utilisateurs et qui permet de gérer des milliers d'agents. Elle ne propose pas, en revanche, d'outils spécifiques pour simplement créer des agents au comportement complexe simulant des acteurs humains.

MASON

MASON [LCRP+05] a été développé comme une alternative à Repast, tout en étant centré sur la gestion de plusieurs centaines de milliers d'agents dans un temps de calcul le plus faible possible. Il s'agit d'une boite à outil Java développée à l'université Georges Mason, open source et disponible gratuitement.

Les agents sont implémentés par des classes Java redéfinissant une classe pré-existante qui est appelée à chaque pas de temps par la plateforme. L'utilisateur définit ensuite une classe représentant la simulation, permettant ainsi d'initialiser les agents et de donner un ordonnancement à l'exécution des agents. En parallèle, une visualisation graphique peut-être développée pour afficher des résultats en 2D et en 3D, là encore en redéfinissant des classes Java pré-existantes.

MASON souffre des mêmes défauts que Repast mais gère beaucoup plus d'agents dans un temps de calcul similaire grâce à la possibilité de définir des méta-agents englobant plusieurs agents similaires. Ces capacités ont permis de réaliser des simulations sur 1000 fourmis devant trouver de la nourriture [PL04] mais aussi sur des sociétés d'humains primitifs pouvant communiquer entre membres d'une même tribu pour trouver de la nourriture ou des abris [CRPL+04].

1.2.2 Présentation de plateformes de simulation

Les frameworks qui enrichissent des langages de programmation génériques sont tournés vers des utilisateurs informaticiens qui sont déjà experts dans les langages de programmation sous-jacent. Ce ne sont pas des outils qui permettent de rapidement définir un modèle sans rentrer dans des subtilités de programmation. Pour contourner ce problème, des plateformes tout en un, comprenant une interface graphique, un compilateur et un interpréteur, ont été proposés. Les trois plateformes présentées dans cette section sont les plus populaires dans la communauté des simulations à base d'agents.

Cormas

Cormas [BBPLP98] est une plateforme libre de droit, disponible gratuitement et développée par le CIRAD avec pour but d'étudier l'interaction entre les ressources naturelles et les sociétés. La plateforme propose son propre environnement de modélisation et de simulation, basé sur VisualWork [Bra15], et s'appuie sur le langage SmallTalk.

La définition d'un modèle se fait en trois étapes distinctes : la définition des agents, les données d'initialisation et d'ordonnancement et la visualisation. Toutes ces étapes sont directement intégrées à la plateforme, sans qu'il y ait besoin de redéfinir d'autres fichiers dans d'autres langages.

Les agents sont définis à partir de classes SmallTalk pré-existantes. Un agent est, dans Cormas, soit spatial, soit social, soit passif. Ces trois types déterminent les méthodes auxquelles l'agent a accès. Ainsi, une case de l'environnement pourra être de type spatial quand un agent actif sera de type social. Dans ce contexte, un agent social n'est pas nécessairement un agent simulant un acteur humain. Une fois le type sélectionné, l'utilisateur définit les méthodes et caractéristiques de l'agent, aidé par l'interface qui offre de nombreuses possibilités d'auto-complétion, de suggestion de mot ou encore de mise en forme automatique du code. Le comportement d'un agent peut aussi être défini via à un diagramme d'activité réalisé graphiquement, limitant ainsi le nombre de lignes de code à écrire.

L'utilisateur doit ensuite définir les données utilisées pour initialiser le modèle ainsi que la méthode à appeler à chaque pas de temps. Cela se fait encore une fois via l'interface graphique de la plateforme. Enfin, des éléments de visualisations peuvent être définis, là encore via l'interface graphique.

Cormas a initialement été créée pour étudier l'évolution des ressources affectées par une présence humaine. Récemment, la plateforme a pris une nouvelle direction en mettant l'accent sur la modélisation collaborative [BBLPB16]. La facilité d'utilisation de la plateforme, assurée par sa dimension tout-en-un contenant sa propre interface graphique, son propre compilateur et son propre interpréteur, ainsi que l'utilisation intensive de l'interface graphique pour suppléer à l'écriture de code informatique, a permis de réaliser ses objectifs. La plateforme est aujourd'hui utilisée sur des cas de gestion de ressources à différentes échelles mais aussi dans des jeux de rôles avec les populations locales. En contrepartie, il n'y a aucun outil spécialement dédié à donner un comportement humain aux agents, le modélisateur doit créer l'ensemble du comportement en partant de rien.

NetLogo

NetLogo [WE99] est une plateforme de modélisation et de simulation à base d'agents, disponible gratuitement et en open source, développée pour la recherche et l'éducation. NetLogo désigne à la fois la plateforme de simulation, l'environnement de modélisation fourni et le langage de modélisation imposé.

Le développement d'agents passe par la redéfinition d'une des trois classes génériques. Soit les agents sont mobiles et sont considérés comme des "tortues", soit ils forment l'environnement dans lequel vont évoluer les "tortues", et dans ce cas ils sont considérés comme des "patchs",

soit ils représentent des liens entre les "tortues" et sont considérés comme des "links". Le modélisateur définit le comportement de ses agents en indiquant des procédures à suivre puis indique l'ordonnancement de l'exécution de ces procédures lors de la simulation.

La philosophie première de NetLogo est de fournir une plateforme tout-en-un, contenant l'interface, le compilateur et l'interpréteur ainsi que de très nombreux modèles exécutables, contenus dans une bibliothèque alimentée par la communauté. Les modélisateurs sont alors invités à s'appuyer sur les modèles existant en les modifiant, même s'ils ont toujours la possibilité de définir un modèle en partant de rien.

NetLogo est codé en Java pour assurer sa portabilité sur le maximum de systèmes possibles. Aussi, il est possible d'ajouter des composants à la plateforme en les développant directement en Java, ce qui a permis d'ajouter une architecture inspirée du modèle BDI présenté en section 1.2.3 à la plateforme [SKS08] dans un but éducatif.

La plateforme NetLogo est simple à prendre en main et permet de rapidement réaliser le prototype d'un modèle à base d'agents. Si elle est à ce jour la plateforme de simulation la plus populaire, c'est grâce à sa bibliothèque contenant plus de 150 modèles d'exemples directement exécutables. En peu de temps, un utilisateur novice peut installer la plateforme, lancer un modèle avec un résultat graphique puis accéder au code de ce modèle pour le modifier et ainsi apprendre la syntaxe du langage imposé. NetLogo est utilisé sur tous types de simulations et s'appuie sur sa communauté pour évoluer et proposer des architectures comportementales pré-construites utilisables par tout le monde. Malgré tout, il n'y a, pour le moment, aucuns outils permettant de définir simplement un agent simulant un humain possédant des capacités sociales ou affectives.

GAMA

GAMA [GTG+13] est la plus récente des plateformes présentées dans cette section. Elle est disponible gratuitement, open-source, développée en Java pour être le plus portable possible. GAMA a la volonté d'être simple d'accès, à l'image de NetLogo, tout en proposant des outils permettant de développer simplement des agents complexes et en nombre, comme cela peut-être fait avec MASON ou Repast. Ainsi, comme NetLogo, GAMA est une plateforme tout-en-un qui possède son propre langage de programmation, le GAML. Enfin, GAMA est tournée vers la communauté des simulations sociales en proposant des outils permettant d'intégrer dans les simulations des systèmes d'information géographiques (SIG) ou encore des données statistiques pour initialiser les modèles.

Dans GAMA, il n'est pas imposé de redéfinir des classes précises en fonction du type d'un agent. Tous les éléments de la simulation, ainsi que la simulation elle-même, sont considérés par la plateforme comme des agents lors de l'exécution. Lors du développement d'un modèle, l'utilisateur doit définir au minimum un agent global qui va avoir le contrôle sur tous les autres agents, ainsi qu'un agent "experiment" qui va contrôler la partie visualisation de la simulation. Pour les autres agents, le modélisateur est libre de les définir comme il le souhaite et peut s'appuyer sur de nombreuses fonctions et architectures comportementales directement implémentées.

La philosophie de GAMA est de développer chaque type d'agent indépendamment du reste de la simulation. C'est ensuite le simulateur qui se chargera d'activer les agents un à un, avec différentes options d'ordonnancement possibles. Un agent est composé de caractéristiques, d'un bloc d'initialisation, d'un bloc définissant son apparence visuelle et de réflexes créant son comportement. Lors de son activation, un agent va activer l'ensemble de ses réflexes, un à un en fonction d'une condition booléenne attachée à chaque réflexe. Une fois que l'agent a activé l'ensemble de ses réflexes, la simulation reprend la main et active l'agent suivant dans son ordonnanceur. Lorsque tous les agents ont été activés, un pas de temps est passé et les agents sont de nouveau activés un à un.

GAMA possède les même points forts que NetLogo tout en ajoutant de nombreux outils permettant de complexifier simplement les simulations. Ainsi, il est possible de développer des visualisations en 3D, d'inclure des propriétés physique aux objets, d'inclure des équations différentielles et leur résolution mais aussi d'utiliser différentes architectures comportementales permet-

tant de mieux cadrer les actions de l'agent. Dans cette optique, les prémisses d'une architecture basée sur le modèle BDI présenté en section 1.2.3 ont été développés pour permettre la modélisation d'agents simulant des acteurs humains [TBC+16]. Tous ces outils permettent d'aborder toutes sortes de simulations, qu'il s'agisse de réseaux sous la forme de graphes ou de système s'appuyant sur des données géographiques.

1.2.3 Comportements des agents dans les plateformes de simulation

L'utilisation de plateformes dédiées à la simulation permet à la fois de faciliter la définition de modèle à base d'agents mais aussi d'assurer une valeur scientifiques aux études menées. Les résultats obtenus sont reproductibles et les modèles peuvent être étudiées en eux-même.

Pour autant, les plateformes ne contraignent en rien le comportement des agents. Il est alors intéressant d'ajouter des architectures comportementales à ces plateformes pour donner un cadre au comportement de l'agent. Dans cette section, les approches comportementales offertes aux modélisateurs par les plateformes de simulation à base d'agent présentées précédemment sont détaillées et commentées.

Comportements sans cadre

Cormas, Repast et MASON n'offrent aucune architecture comportementale pour le développement d'agents. Les modélisateurs sont donc libres de donner le comportement qu'ils souhaitent aux agents, sans aucune contrainte structurelle. Le comportement est défini par une programmation de bas niveau, sans que l'utilisateur ne puisse s'appuyer sur des processus décisionnaires déjà implémentés ou des concepts permettant de manipuler une information de haut niveau.

Dans les faits, cela conduit les modèles à s'appuyer sur des comportements simples venant des SMA en suivant le principe KISS (Keep It Simple, Stupid) [Axe97] qui indique que le comportement des agents doit rester explicable à tout instant. Cela signifie avoir une description simple des agents et de leur comportement qui se résume le plus souvent à un ensemble de règles du type "si-alors-sinon". L'agent récupère des informations de son environnement qui servent de condition de déclenchement pour un ensemble d'actions.

Comportements selon une architecture

Ce comportement simpliste d'agents simulant des humains a été remis en question par la communauté [Sun07] car un humain, dans une situation complexe, ne fait pas que réagir à son environnement, il prend une décision à la suite d'un processus cognitif. Pour mettre en œuvre ce comportement cognitif, NetLogo et GAMA proposent à leurs utilisateurs la possibilité d'utiliser l'implémentation d'une architecture comportementale [Occ03] fondée sur le modèle de cognition BDI [Bra87] en plus de la possibilité de définir le comportement simplement par des règles "si-alors-sinon".

Le principe de base du modèle de décision BDI est de considérer qu'un agent possède trois bases d'états mentaux sur lesquels il va s'appuyer pour prendre une décision : ses croyances, ses désirs et ses intentions. Les croyances représentent ce que l'agent sait de son environnement, les désirs représentent les états que l'agent souhaite atteindre et les intentions représentent les états que l'agent s'emploie à atteindre par le biais de plans d'action. Le moteur de raisonnement va alors faire le lien entre les plans d'action possibles en fonction du contexte, c'est à dire des informations connues de l'agent, et les désirs de l'agent pour définir des intentions.

Ce modèle théorique a été implémenté dans la communauté des SMA selon le modèle PRS (Procedural Reasoning System) [Mye97] dont une représentation est proposée par la Figure 1.1. Par abus de langage, il est aujourd'hui question d'architecture BDI. Dans cette architecture, les informations venant de l'environnement sont transformées en croyances et viennent mettre à jour la base des croyances de l'agent. La prise de décision va alors consister à mettre en relation les intentions de l'agent, ses désirs et le contexte local afin de trouver le plan d'action qui répond au plus de conditions.

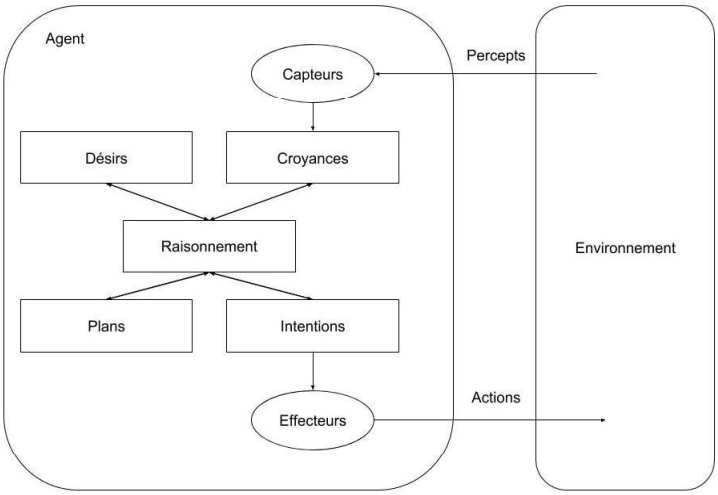

FIGURE 1.1 – Schéma d'une architecture BDI se basant sur PRS pour le comportement d'agents logiciels

Dans une architecture BDI, un agent va persister dans son intention tant qu'elle n'est pas atteinte et qu'elle est toujours atteignable. Cela signifie que l'agent va avoir une inertie dans son comportement, mais aussi une motivation plus profonde que simplement répondre à un changement dans l'environnement.

L'implémentation réalisée dans NetLogo [SKS08] est une version simplifiée d'une architecture BDI, dans un but éducatif. L'agent a accès à une pile d'intentions formées par le nom d'une action et une condition de fin de cette intention. Le cycle de raisonnement de l'agent est le suivant :

1. L'agent sélectionne comme intention courante l'intention la plus ancienne dans sa pile des intentions.
2. l'agent exécute l'action liée à son intention courante.
3. Si la condition de fin n'est pas atteinte, l'intention courante est conservée et le moteur repartira de l'épate 2 au pas de calcul suivant.
4. Si la condition de fin est atteinte, l'intention courante est supprimée de la pile des intentions. Le moteur recommencera le cycle à l'étape 1 au temps de calcul suivant.

En plus d'une pile d'intentions, l'agent a accès à une base de croyances formées par un nom et une valeur qui permet de stocker des information de haut niveau. Pour autant, ces croyances ne sont pas directement prises en compte dans le moteur de raisonnement.

L'architecture cognitive implémentée dans GAMA [TBC+16] est plus complète en proposant une base de croyances, une base de désirs ainsi qu'une base d'intention. Le cycle de raisonnement de l'agent, représenté dans la figure 1.2, est le suivant :

1. L'agent perçoit son environnement, transformant les informations de l'environnement en croyances.
2. L'agent applique des règles d'inférence, définies par le modélisateur, permettant d'ajouter ou de retirer des désirs en fonction des croyances nouvellement acquises.
3. Le moteur cognitif sélectionne le désir de plus haute priorité pour en faire l'intention courante.
4. Un plan d'action répondant à l'intention courante et dont les conditions d'activation sont réunies, défini par le modélisateur, est sélectionné et exécuté.
5. Si le plan n'a pas atteint sa condition de fin, il est conservé pour pour le pas de calcul suivant.
6. Si l'intention n'a pas été atteinte, elle est conservée pour le pas de temps suivant.

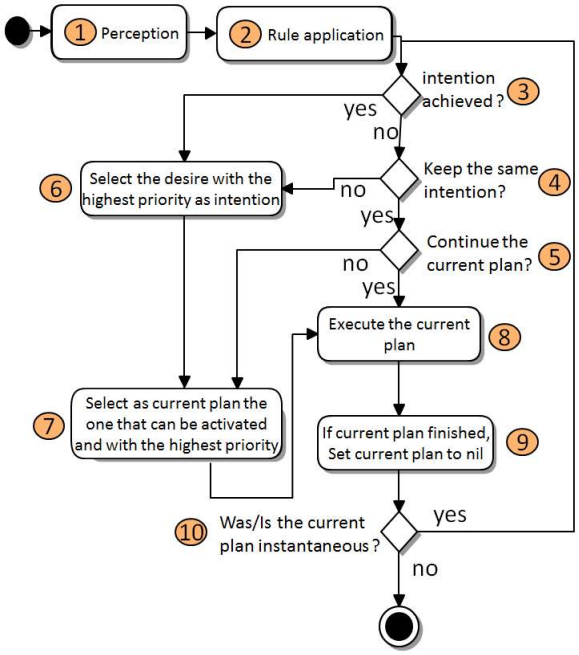

FIGURE 1.2 – Diagramme d'activité de l'architecture BDI implémentée dans GAMA, tiré de [TBC+16]

Cela signifie que l'utilisateur doit définir les perceptions de l'agent, ses règles d'inférences ainsi que les plans d'action. Plusieurs plans d'action peuvent être définis pour répondre à une même intention, offrant différents comportements à des agents ayant la même intention mais n'ayant pas les mêmes croyances. Cette architecture BDI est l'architecture cognitive la plus complexe implémentée dans une plateforme de modélisation et de simulation à base d'agents.

1.3 Les architectures comportementales pour les agents logiciels

L'ajout d'un comportement cognitif n'est qu'une première étape pour rendre les agents sociaux plus proches des humains simulés comme l'indique le principe EROS (Enhancing Realism Of Simulation) [Jag17], en opposition du principe KISS. La volonté de la communauté est aujourd'hui de doter les agents simulant les êtres humains de capacités sociales, utilisées par les personnes simulées pour prendre des décisions. Cela signifie ajouter des compétences sociales et affectives de haut niveau, simulant les compétences humaines dans ces domaines.

Cette section a pour but de montrer quelques approches permettant de développer un comportement agent complexe mais qui n'ont, à l'heure actuelle, pas été intégrées dans les plateformes de simulation. De ce fait, ces travaux ne sont pas majoritairement utilisés par la communauté de la simulation sociale.

Les notions issues des sciences sociales évoquées dans cette section telles que la cognition, les normes, les émotions, la personnalité, la contagion émotionnelle ou encore les relations sociales sont définies en détails dans le chapitre 2.

1.3.1 Architectures cognitives

En parallèle de l'implémentation directe d'une architecture BDI dans les plateformes NetLogo et GAMA, comme expliqué en section 1.2.3, il a été proposé un framework permettant de connecter une plateforme de simulation à une plateforme comportementale de développement d'agent BDI [SPL16]. L'idée de ce travail est d'offrir un cadre dans lequel une plateforme de simulation délègue la prise de décision de ses agents simulant des acteurs humains à une plateforme spécialisée dans le modèle BDI.

Le défi de cette approche est de connecter des plateformes qui ne fonctionnent pas de la même façon et n'utilisent pas les mêmes langages de programmation. Pour ce faire, la plateforme de simulation a la main. Lorsqu'un agent simulant un humain doit prendre une décision, les informations de perception sont encapsulées dans un type de donnée qui est passé à l'architecture BDI extérieur qui s'occupe alors de sélectionner une des actions permises par la plateforme de simulation et renvoie le résultat. L'action est alors exécutée dans la plateforme de simulation.

Les auteurs expliquent avoir appliqué ce framework à différentes association, notamment en liant les plateformes Jack et MATSim [BRM+09] dans l'étude d'une évacuation pour un incendie de brousse, en liant les plateformes Jadex et Repast pour étendre un modèle de démonstration ou encore en liant les plateformes Gorite [Rön07] et MATSim pour la gestion d'un service de taxis. Cette approche est tournée vers une communauté d'informaticien puisqu'il faut maîtriser deux langages de programmation, deux plateformes mais aussi la communication entre ces plateformes, ce qui nécessite de bonnes connaissances en programmation.

Il existe d'autres architectures cognitives que les travaux fondés sur le modèle BDI décrit en section 1.2.3. Ainsi, l'architecture SOAR [LNR87], détaillée en section 2.1.2, propose de donner à l'agent une mémoire à courte durée et une mémoire à longue durée. L'agent utilise sa mémoire à courte durée pour traiter une perception et y répondre puis évalue cette réponse pour apprendre de son comportement et stocker le résultat dans sa mémoire à long terme. Cette mémoire à long terme agira dans le futur sur la mémoire à court terme afin de changer la prise de décision dans un contexte similaire.

ACT-R (Adaptative Control of Thought-Rational) [BA98] est une autre architecture cognitive, représentée par la figure 1.3, qui propose de séparer les états mentaux de l'agent en deux catégories : les états déclarés représentant les faits et les états intentionnels représentant les plans d'action.

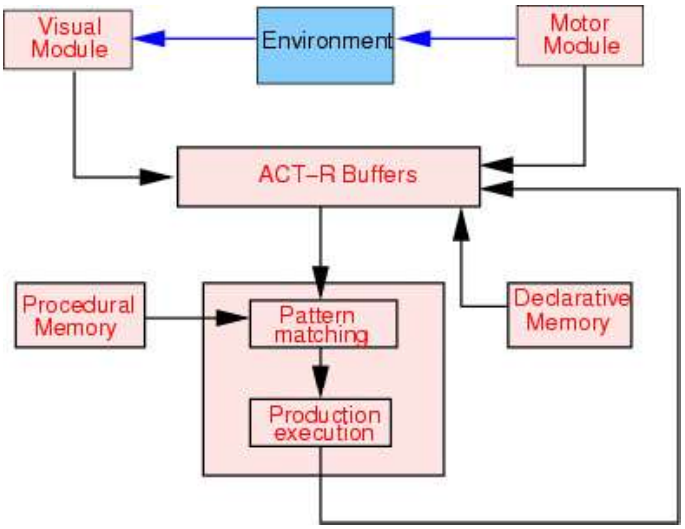

FIGURE 1.3 – Schéma de l'architecture ACT-R, tiré du site officielle http ://act-r.psy.cmu.edu/

Un agent fonctionnant avec l'architecture ACT-R a le cycle de raisonnement suivant :

1. La perception de l'environnement met à jour la mémoire déclarative de l'agent.
2. Le processus de raisonnement central trouve un plan d'action en fonction de la mémoire déclarative de l'agent.
3. Ce plan est stocké dans un buffer.
4. Si le plan est exécutable, le buffer active des effecteurs qui vont agir sur environnement en suivant le plan.
5. Si le plan n'est pas exécutable, le buffer va refuser le plan, stocker cette information dans la mémoire déclarative et relancer le raisonnement à l'étape 2.

L'architecture CLARION (Connectionist Learning with Adaptative Rule Induction ON-line) [SMP01] va plus loin dans le même esprit en divisant le raisonnement de l'agent en quatre sous-systèmes, comme le montre la figure 1.4 : le sous-système lié aux motivations, le sous-système méta-cognitif, le sous-système centré sur les actions et le sous-système non centré sur les actions. Chaque sous-système possède des éléments représentés explicitement et des éléments représentés implicitement permettant ainsi à l'agent de prendre plusieurs décisions en même temps, une par sous-module.

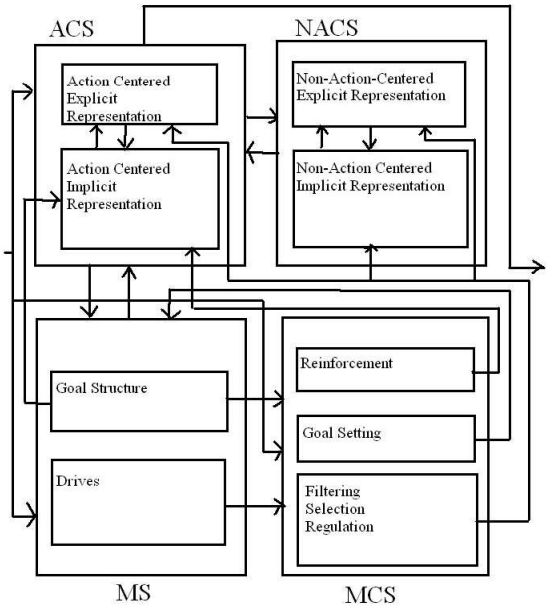

FIGURE 1.4 – Schéma de l'architecture CLARION, tiré de [Sun06]

Ces propositions d'architectures cognitives, plus complexes que les architectures BDI, n'ont, pour l'heure, pas encore été implémentées dans les plateformes de simulations présentées en section 1.2.2. Les raisons sont multiples : CLARION est une architecture théorique qui n'a pas été implémentée alors que SOAR et ACT-R sont gourmands en temps de calcul si bien qu'ils n'ont été utilisés que sur des cas impliquant un petit nombre d'agents [JTLR93] [BL06].

1.3.2 Architectures normatives

Une autre façon de donner un comportement aux agents, qui n'a pas été exploré dans les plateformes présentées en section 1.2, est de passer par un système normatif. Un SMA normatif est alors un ensemble d'agents dont le comportement suit un ensemble de normes pouvant évoluer en fonction du comportement des agents [BVDTV06].

Hubner [HSB02] et Dignum [Dig99] ont proposé une approche bas niveau des SMA normatifs en s'appuyant sur la logique déontique [MW$^+$93] permettant de définir précisément le comportement d'agents en fonction d'obligation, de permissions et d'interdictions imposées par le système. Dans les deux cas, le but est de permettre la définition, en même temps, de la structure et du fonctionnement du système normatif. Hubner prend l'exemple d'une équipe de football. La structure de l'équipe permet de définir des rôles pour les agents, ainsi que des liens entre ces rôles indiquant si les agents peuvent communiquer ou non. Le fonctionnement du système repose sur des missions composées de buts organisés en plan. La logique déontique est alors utilisée pour faire le lien entre la structure et le fonctionnement du système en indiquant les obligations et les permissions de chaque rôle vis à vis des missions préalablement définies.

Des approches de plus haut niveau proposent d'inclure le concept de normes dans des architectures agents, comme c'est le cas de l'architecture EMIL-A [ACCP07] représentée par la figure 1.5.

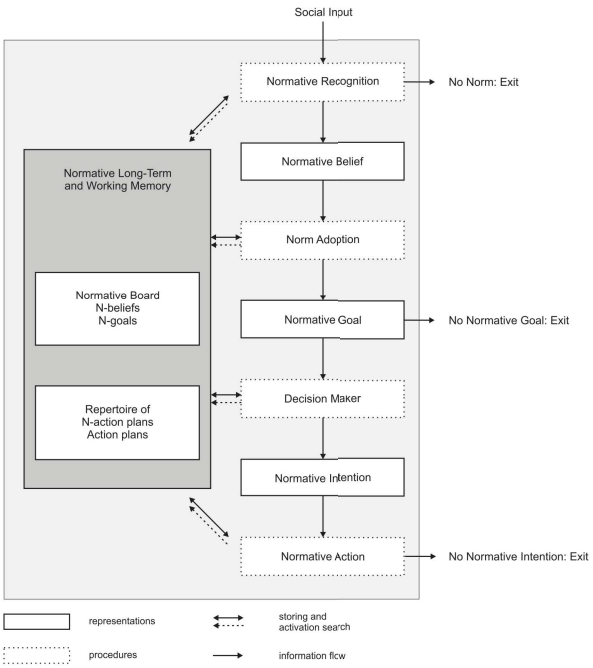

FIGURE 1.5 – Schéma de l'architecture EMIL-A, récupérée de [BG14], reproduite à partir de [ACCP07]

Un agent agissant selon l'architecture EMIL-A aura le cycle de raisonnement suivant :

1. L'agent perçoit les normes venant de l'environnement et en crée des croyances (par exemple : *je crois qu'il est interdit de ...*).
2. Une phase d'adoption de la norme permet à l'agent de choisir s'il crée un but à partir de la norme ou non. Dans ce second cas, la norme est considérée comme violée.
3. Les nouveaux buts créés sont alors comparés avec les buts pré-existants afin de définir l'intention à réaliser.
4. L'intention peut alors être exécutée ou abandonnée, conduisant dans le second cas à une violation de la norme.

Dans le même esprit, l'architecture NoA [Kol05], représenté par la figure 1.6, propose un cycle de raisonnement similaire en s'appuyant sur une définition plus large du concept de norme. Ainsi, une norme peut représenter une obligation, une permission ou une interdiction mais aussi une indication de pouvoir, un privilège ou une immunité. Les normes sont ainsi contextualisées et l'agent conserve en mémoire l'ensemble de ces données sous la forme d'un état normatif explicite. L'autre différence majeure est que les plans d'action renseignent les normes qu'ils suivent en plus de leurs conditions d'activation. De cette façon, l'ensemble du comportement peut se décrire comme une suite de normes à suivre, donnant moins d'importance aux concepts de désirs et d'intentions.

Une autre façon de réaliser des architectures comportementales normatives consiste à ajouter une gestion des normes à une architecture cognitive. L'architecture BOID [BDH+01] propose ainsi d'ajouter la notion d'obligation à une architecture BDI. Ces obligations, imposées par une autorité, sont mises en concurrence avec les désirs de l'agent. Le moteur de raisonnement classe ces désirs et ces obligations en fonction d'une valeur de priorité et sélectionne l'état mental de plus haute priorité pour en faire une intention.

BOID a servi de base à l'architecture BRIDGE (Beliefs Response Intention Desire Goal Ego) [DDJ08] qui propose d'ajouter différents composants à une architecture BDI traditionnelle, comme le montre la figure 1.7. Le but est d'ajouter une conscience sociale à l'agent, ainsi qu'une conscience de lui-même. Cette approche propose aussi de différencier les buts et les intentions de l'agent, les buts découlant des désirs quand les intentions représentent les plans possibles pour réaliser les buts.

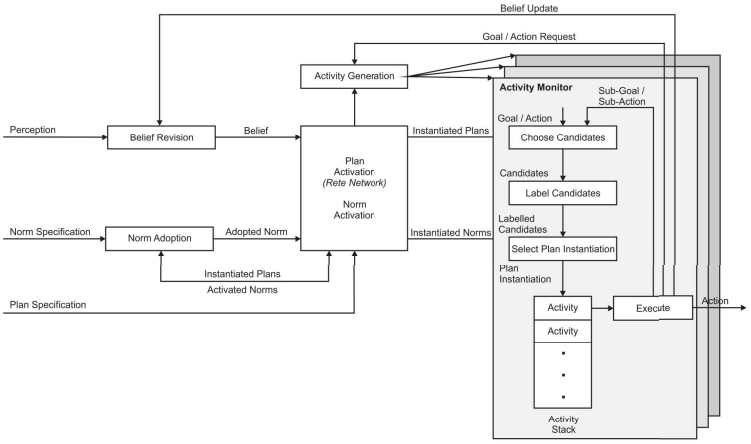

FIGURE 1.6 – Schéma de l'architecture NoA, extrait de [BG14], reproduite à partir de [Kol05]

Le cycle de raisonnement de l'architecture BRIDGE fait fonctionner tous les modules en même temps ce qui oblige à prendre en compte les perceptions de façon continue. Dans le même temps, un composant de personnalité, inclus dans le module *Ego*, permet de transformer les perceptions en croyances. Ces croyances agissent ensuite comme des filtres pour créer des buts à partir des désirs. Le module *Response* permet enfin d'ordonner les buts, en fonction de l'urgence d'une action à exécuter pour aider à la sélection d'une intention.

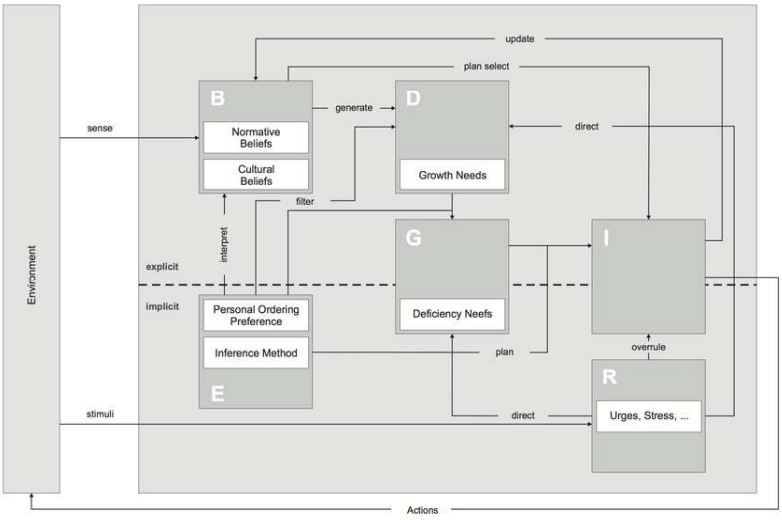

FIGURE 1.7 – Schéma de l'architecture BRIDGE, récupérée de [BG14], reproduite à partir de [DDJ08]

BRIDGE représente l'architecture proposant le plus de dimensions sociales et psychologique se basant sur BDI. Néanmoins, elle reste théorique et n'a, pour le moment, jamais été implémentée. Sa mise en place demanderait de grands efforts dans la représentation formelle des concepts mais aussi dans la description précise des processus permettant la prise de décision et créant les liens entres les modules.

1.3.3 Architectures émotionnelles

Une autre approche possible pour rendre le comportement d'agents sociaux plus proche du comportement humain est l'intégration d'émotions [Dam94]. Plusieurs solutions ont été proposées, chacune basée sur son propre mécanisme [BTVA18]. Les approches les plus simples proposent de représenter une émotion par une valeur numérique qui évolue dans le temps, par exemple [TFB+11], quand les approches les plus complexes proposent une représentation symbolique avec

une évolution liée à des règles, comme avec [HJC03]. Le problème de ces travaux est qu'ils ne s'appuient pas sur des théories émotionnelles mais qu'ils considèrent les émotions d'un agent comme un paramètre supplémentaire.

L'architecture eBDI (Emotion BDI) [JVH07], représentée par la figure 1.8, propose d'ajouter une composante émotionnelle au raisonnement BDI pour créer une architecture émotionnelle. De cette façon, des émotions sont créées, en fonction d'une théorie choisie lors de l'implémentation, en même temps que les croyances de l'agent sont mises à jour par la perception. Ces émotions peuvent ensuite influencer la prise de décision, à la convenance du modélisateur.

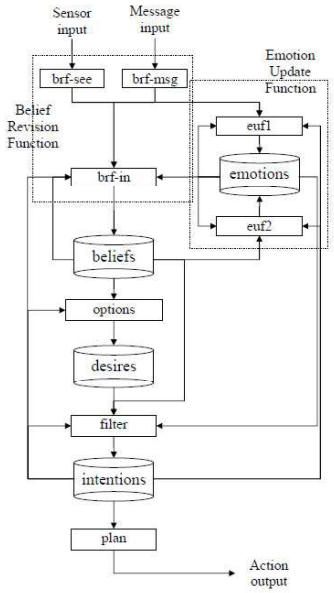

FIGURE 1.8 – Schéma de l'architecture eBDI, tirée de [JVH07]

EMA [GM04]représentée par la figure 1.9, est au contraire une architecture mettant les émotions au centre du processus de décision de l'agent. EMA repose sur la théorie de l'évaluation cognitive des émotions [SL$^+$90] qui consiste en deux étapes : une première phase d'évaluation des états mentaux après perception de l'environnement permet de créer des émotions puis une seconde phase permet d'adapter le comportement de l'agent à son état émotionnel, en fonction de différentes stratégies. Ainsi, ce sont les émotions de l'agent qui dirigent son comportement.

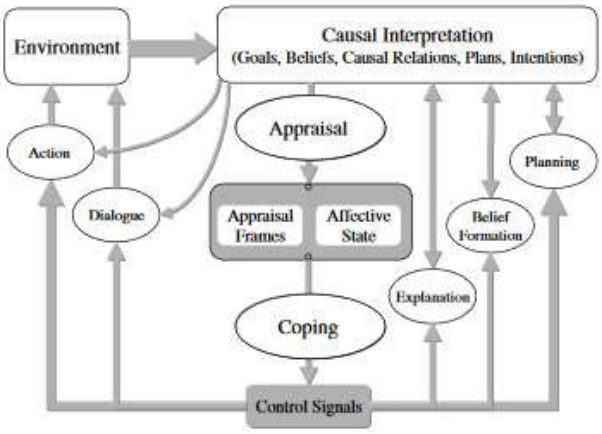

FIGURE 1.9 – Schéma de l'architecture EMA, tirée de [GM04]

DETT (Disposition, Emotion, Trigger, Tendency) [VDPBB$^+$06], représentée par la figure 1.10, est un autre exemple d'architecture émotionnelle pour la simulation sociale. Là encore, la phase de création des émotions est détachée de leur utilisation dans la prise de décision. Le modélisateur

définit des *déclencheurs*, c'est à dire des perceptions qui donneront lieu à des émotions. Chaque agent possède un module de *disposition* représentant une personnalité et expliquant pourquoi deux agents peuvent avoir des émotions différentes face à la même perception et enfin, le module de *tendance* contient les règles indiquant comment les émotions influencent le comportement. Ainsi, cette architecture laisse au modélisateur la liberté de programmer son propre système émotionnel, couplé à un moteur cognitif basé sur le modèle BDI.

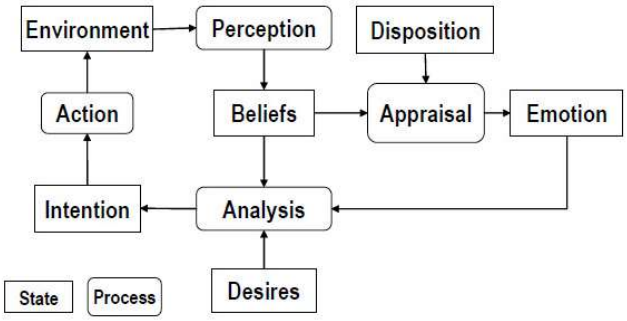

FIGURE 1.10 – Schéma de l'architecture DETT, tirée de [VDPBB+06]

Une autre approche est de considérer l'état émotionnel de l'agent comme un tout et non de considérer toutes ses émotions une à une. PAD (Pleasure Arousal Dominance) [RCN+16] considère l'état émotionnel d'un agent comme un point dans un espace en trois dimensions. Ce modèle s'appuie sur de précédents travaux travaillant uniquement avec les dimensions de plaisir (Pleasure en anglais), indiquant si l'émotion est positive ou négative, et de conscience (Arousal en anglais), indiquant le degré de d'activité de l'émotion [SG10]. Cet état émotionnel évolue en fonction des perceptions et tend à revenir vers un état neutre, lui donnant une dynamique continue.

Rassembler l'ensemble des émotions d'un agent sous une seule valeur d'état émotionnel a conduit à travailler avec la notion d'humeur. Si une émotion est un phénomène ponctuel, qui ne dure que quelques minutes, l'humeur est une notion qui évolue plus lentement, sur plusieurs heures [For92]. L'humeur peut donc être utilisée dans des simulations de systèmes évoluant sur des durées longues, servant de passerelle entre des émotions ponctuelles et des décisions prises sur le long terme, ce qui est proposé par le modèle ALMA [Geb05]. Le cycle de raisonnement de l'agent suit les étapes suivantes :

1. L'agent crée des émotions en s'appuyant sur la théorie OCC de l'évaluation cognitive des émotion [OCC90].
2. Les émotions créées influencent l'état émotionnel de l'agent, représenté dans un espace PAD fermé entre -1 et 1 pour chacune de ses dimensions.
3. En fonction de son emplacement dans l'espace PAD, l'état émotionnel exprime une des huit humeurs possibles.
4. L'agent prend une décision en fonction de son humeur.

Le modèle ALMA représente le travaille le plus complet sur le comportement émotionnel en prenant en compte aussi bien l'évaluation des actions à court terme que l'humeur à long terme qui influence le comportement. Pour autant, la notion d'humeur n'est intéressante que sur un temps long. ALMA est utilisé sur un agent conversationnel animé qui reste activé sur une longue période. Dans le cadre de simulations sociales, si la situation étudiée ne court que sur quelques minutes, l'humeur d'une personne n'a pas le temps d'évoluer significativement pour être une dimension prise en compte dans la simulation.

1.3.4 Comportements mêlant plusieurs dimensions psychologiques et sociales

Simuler des acteurs humains implique d'inclure dans la définition du comportement des agents au moins une des notions évoquées dans la section 1.3 et si possible plusieurs. Certains chercheurs

ont donc proposés des comportements agents fondés sur plusieurs dimensions cognitives, sociales et affectives, sans pour autant les inclure dans une architecture comportementale qui serait détachée de tout cas d'utilisation.

Ochs [OSC09b] propose d'ajouter aux émotions et à la cognition des relations sociales entre les agents, en se basant sur le modèle des relations interpersonnelles [Sve00]. L'architecture proposée lie l'évolution des relations sociales aux émotions de l'agent, émotions créées après évaluation cognitive de l'environnement. Ces relations sociales peuvent ensuite impacter la prise de décision des agents.

Sur le même modèle, Lhommet[LLB11] propose de combiner la cognition, les émotions et la contagion émotionnelle. Pour ce faire, une personnalité est ajoutée aux agents en suivant le modèle OCEAN [MJ92]. Le cycle de raisonnement de l'agent reste classique, avec une perception de l'environnement, une création des émotions et une modification des émotions créées en fonction des émotions des agents perçus. Tous les paramètres de cette contagion sont initialisés en fonction de la personnalité des agents.

Certains travaux se sont concentrés sur la contagion émotionnelle comme base du comportement pour des agents sociaux. Le modèle ASCRIBE (Agent-based Social Contagion Regarding Intentions, Beliefs and Emotion) [BDM+09] décrit précisément la contagion émotionnelle entre deux agents. Dans ce modèle, les émotions sont vues comme des valeurs évolutives dans le temps. La contagion émotionnelle va permettre de faire se propager une émotion dans une foule, émotion qui décidera d'un comportement. ASCRIBE a été appliqué aux éléments BDI d'un agent pour donner le modèle IMPACT [vdWFR+17]. L'ensemble des éléments constituant le raisonnement d'un agent sont des valeurs numériques qui évolues et s'harmonisent en fonction des autres agents perçus. Ainsi, certains agents vont avoir des croyances à forte valeur sur des événements qu'ils n'ont pas perçus, simplement parce qu'un agent les aura "contaminé" avec cette croyance.

Une toute autre approche est proposée par [CC+16] en plaçant les normes sociales comme force motrice du comportement cognitif d'un agent. Cette réflexion vient du constat que la cognition seule ne peut expliquer le comportement de quelqu'un dans un groupe. Les normes sociales sont alors utilisées pour cadrer les désirs et les buts d'un agent agissant dans un groupe. Les recherches en systèmes normatifs [CDJT99] menés dans les SMA sont projetées sur un contexte de simulation sociale où les agents ne doivent plus être optimaux envers une solution mais représenter au mieux des humains.

1.4 Conclusion

Ce chapitre montre qu'il existe de nombreuses approches pour rendre les agents sociaux plus proches, dans leur comportement, des humains qu'ils sont sensés simuler. Une façon de rendre ces agents sociaux plus crédibles est de s'appuyer sur les recherches en psychologies et en sociologie, qui indiquent que le comportement humain possède une part de cognition mais dépend aussi de notions comme les émotions, les relations sociales, les normes, la personnalité ou encore la contagion émotionnelle. Ce chapitre montre également que différents travaux ont été menés, dans le domaine des SMA, pour créer des agents logiciels implémentables offrant des comportements complexes.

Le tableau 1.1 résume les architectures comportementales existantes, et discutées dans le chapitres, en mettant en avant les composants comportementaux inclus ainsi que l'échelle de leurs applications en termes de nombre d'agents simulés et d'intégration dans des plateformes de simulation.

Le tableau 1.1 montre tout d'abord qu'aucune architecture ne propose en même temps l'ensemble des dimensions psychologiques de la prise de décision. Certaines architectures sont uniquement cognitives (SOAR, ACT-R et Clarion), certaines sont uniquement fondés sur la gestion des normes (EMIL-A et NoA), d'autres ajoutent une gestion des normes à la cognition (BRIDGE), certaines architectures sont uniquement émotionnelle (EMA), et d'autres ajoutent des émotions à la prise de décision (EMA, DETT, ALMA). Seule l'architecture PRS, basée sur le modèle BDI, a été

TABLEAU 1.1 – Architectures de comportements agents abordés dans le chapitre 1

Architectures	Cognition	Émotions	Normes	Commentaire
PRS [Mye97]	X	-	-	implémente le modèle BDI [Bra87] de la cognition, implémentée dans la plateforme de simulation GAMA [TBC+16]
SOAR [LNR87]	X	-	-	utilisée sur une simulation d'avions de chasse [JTLR93], comprend un système d'apprentissage et de contextualisation sémantique
ACT-R [BA98]	X	-	-	utilisée pour de la coopération entre humain et joueurs virtuelles sur un jeu vidéo [BL06], aucune intégration des dimensions affectives et sociales
Clarion [SMP01]	X	-	-	architecture théorique reposant sur une représentation implicite d'états mentaux donc complexe à implémenter
EMIL-A [ACCP07]	-	-	X	utilisée sur une simulation de personnes attendant dans une queue pour le projet EMIL [ACTP07], résume le comportement de l'agent à une gestion des normes
NoA [Kol05]	-	-	X	non appliquée en simulation, ajoute une mémoire normative à l'agent
BOID [BDH+01]	X	-	X	architecture théorique, ajout du concept d'obligation à une architecture BDI
BRIDGE [DDJ08]	X	-	X	architecture théorique, étend l'architecture BOID avec des modules de personnalité
eBDI [JVH07]	X	X	-	ajout d'émotions à une architecture BDI, utilisée sur simulation de trafic routier [JSL09]
EMA [GM04]	-	X	-	utilisée sur des simulations d'un seul agent militaire [GM05], l'ensemble du comportement de l'agent est régi par ses émotions
DETT [VDPBB+06]	X	X	-	cas d'exemple sur le comportement de soldats [VDPBB+06], le système émotionnel conditionne la cognition
ALMA [Geb05]	X	X	-	utilisée sur des agents conversationnels animés, utilisation de l'humeur comme une couche émotionnelle à longue durée pour des prises de décisions sur des temps longs

modifiée pour gérer d'un coté des émotions (eBDI) et de l'autre des obligations (BOID). Il manque donc une architecture qui permettrait de définir en même temps des agents cognitifs, affectifs et sociaux.

Le deuxième enseignement du tableau 1.1 est qu'aucune des architectures présentées n'a été utilisée sur des cas impliquant des centaines d'agents ; seule l'architecture PRS a été mise en place dans une plateforme de simulation. Aussi, comme le montre la section 1.2.3, les plateformes les plus populaires ne disposent au mieux que d'une architecture cognitive, fondée sur le modèle BDI. Si c'est une première étape dans la définition d'agents sociaux, cela reste insuffisant. Développer une architecture comportementale comprenant des dimensions sociales et affectives dans une de ces plateformes permettrait à la communauté des simulations sociales d'utiliser les modèles de comportement avancés pour simuler des acteurs humains.

Pour cela, il est possible de s'appuyer sur les architectures développées dans la section 1.3. Le problème est que ces architectures sont indépendantes les unes des autres et ne sont pas faites pour être compatibles entre elles. De plus, elles ne s'appuient pas toutes sur les mêmes concepts ou les mêmes représentations : l'architecture EMIL-A considère les éléments de cognition comme des normes, les architectures ACT-R et Clarion travaillent avec différents niveaux de mémoire et une représentation partiellement implicite des notions cognitives et l'architecture EMA considère les émotions comme le déclencheur du comportement. Il faut donc unifier les concepts et leur représentation pour proposer une architecture homogène, ne mettant pas en avant une dimensions comportementale particulière.

Cette thèse a pour vocation de proposer une architecture comportementale pour la simulation sociale contenant à la fois un moteur cognitif, ainsi que des dimensions affectives et sociales. Comme le montre la section 1.3.4, les dimensions affectives et sociales ne se limitent pas aux émotions et à la gestion des normes ; des notions comme les relations sociales, la personnalité ou la contagion émotionnelles peuvent être prisent en compte.

L'architecture développée doit être implémentée dans une plateforme de simulation afin de la rendre utilisable par la communauté des simulations sociales. Elle doit aussi être modulaire pour être le plus adaptable possible et ne pas imposer des dimensions comportementales aux utilisateurs. Le but est d'amener les modélisateurs à définir des comportements crédibles dans la simulation d'acteurs humains, en facilitant l'utilisation de la cognition et des dimensions affectives et sociales par le biais d'une architecture déterminant ce comportement, et directement intégrée dans les outils déjà utilisés par les modélisateurs.

Chapitre 2

Les dimensions de la prise de décision

Sommaire

Pour rendre crédible dans leurs comportements des agents simulant des acteurs humains, nous proposons de mimer les processus de prise de décisions humains. Pour cela, il convient d'étudier cette prise de décision chez l'homme, en s'appuyant sur les travaux menés dans les sciences humaines et sociales.

La question du comportement humain a longtemps été réservée au domaine de la philosophie, avant de prendre une autre direction, avec le domaine de la psychologie, au milieu du 19ème siècle. Depuis, plusieurs courants de pensée ont été proposés pour expliquer la prise de décision chez l'Homme.

À la jointure entre le 19ème et le 20ème siècle, la psychanalyse est la première discipline à s'attaquer à l'étude du comportement humain via une méthode scientifique d'expérimentation [FBBB56]. Cela a été suivi par le courant du behaviorisme qui s'est proposé de définir la prise de décision comme une réponse directe à un stimulus perçue [Wat13]. Finalement, c'est le courant cognitiviste [Bru17] qui s'est imposé entre 1950 et 1960 comme le courant majoritaire sur la question du comportement humain, donnant naissance au domaine des sciences cognitives qui met en relation la philosophie, la psychologie, la linguistique, l'anthropologie, les neurosciences et l'intelligence artificielle.

Au milieu des années 1990, une divergence s'est marqué dans la définition d'un comportement cognitif entre sa définition psychologique et son application dans les SMA : pendant que Damasio [Dam94] démontrait l'importance des émotions dans la cognition, les SMA commençaient simplement à introduire des éléments de cognition [RG+95] dans la définition de leurs agents. À partir de là, les émotions, puis les dimensions sociales, ont été considéré comme faisant partie intégrante de la cognition, du point de vue psychologique, alors que la majorité des utilisations en IA distingue les comportements cognitifs (qui tiennent du raisonnement) des comportements émotionnels et sociaux.

Dans ce chapitre, les notions relatives au comportement humain sont traitées sous l'angle de l'intelligence artificielle. Cela signifie qu'une distinction est faite entre la cognition qui se résume ici au raisonnement rationnel, les phénomènes affectifs, qui composent la dimension non rationnelle d'un agent, et les phénomènes sociaux, qui forment les éléments de comportement liés aux autres agents. Le but de ce chapitre est donc de définir précisément, du point de vue informatique, quels sont les composants identifiés par les sciences cognitives dans la prise de décision d'une personne se trouvant dans un contexte social donné.

2.1 Prise de décisions cognitive

Comme indiqué en introduction de ce chapitre, le terme "cognition" ne désigne dans cette section, et dans cette thèse, que les processus de raisonnement rationnel permettant la prise de décision d'un acteur humain. Pourtant, du point de vue psychologique, la cognition est majoritairement vue comme l'ensemble des fonctions de l'esprit permettant de construire une représentation opérationnelle de l'environnement à partir de perceptions dans le but de nourrir le raisonnement mais aussi d'aider à la décision d'actions. Cette vision psychologique de la cognition représente une définition plus large que celle qui est considérée dans cette thèse.

L'ajout de processus de raisonnement rationnels est une première étape pour rendre le comportement d'agent informatique plus proche du comportement humain [BG14]. Cela offre un processus de prise de décision basé sur des concepts de haut niveau qui permettent à l'agent de ne pas simplement répondre à un changement dans son environnement mais de perdurer dans ses tâches, comme le ferait un acteur humain.

Cette idée de la cognition provient de travaux menés en psychologie, présentés dans la première sous section. Par la suite, nous regardons la façon dont ces travaux ont été traduits dans l'intelligence artificielle, et plus particulièrement dans les SMA. Enfin, des formalisations du modèle BDI sont présentées.

2.1.1 Introduction à la psychologie cognitive

La psychologie cognitive est une branche de la psychologie qui découle des études menées pour comprendre les processus mis en place par un être humain pour raisonner et prendre une décision. Cette question a longtemps été réservée au débat philosophique, jusqu'à ce que des observations sur des expérimentations amènent à l'essor de la psychologie comme discipline à part entière avec pour ambition d'étudier les processus du comportement humain par le biais de théories testées dans le cadre d'expérimentations.

Le premier mouvement important de la psychologie a été la psychanalyse, symbolisée par les avancées de Freud [FBBB56]. La psychanalyse se propose d'expliquer le comportement humain en étudiant l'inconscient. Le principe est de dire qu'une pensée ou un raisonnement n'est pas un acte anodin. Il y a donc, selon la psychanalyse, un enchaînement logique, potentiellement inconscient, de causes menant à une certaine pensée.

Assez rapidement, la psychanalyse a subi des critiques, principalement parce que l'ensemble du comportement humain était ramené à des causes inconscientes profondes, ce qui ne semblait pas forcément adapté à des situations simples comme la prise de décision concernant une action élémentaire tel que manger. Ces critiques ont mené a la création du behaviorisme [Wat13] qui se propose d'étudier le comportement observable et non les processus inconscients qui ne peuvent être que supposés.

Le behaviorisme prend ses racines dans les études sur le conditionnement d'animaux menées par Pavlov [Pav10]. Ces travaux ont permis de montrer que des animaux arrivent à avoir un comportement donné en fonction d'un stimulus qui leur est délivré, même s'il n'y a pas de lien de cause à effet a priori entre ces éléments. L'exemple le plus célèbre montre que l'on peut associer un son à l'activité de manger chez un chien. Une fois l'association réalisée, le chien sera conditionné pour saliver dès lors qu'il entendra ce son, quand bien même aucune nourriture ne lui serait présentée.

Watson [Wat13] a alors appliqué un principe similaire pour expliquer le comportement humain, donnant naissance au behaviorisme et qui se résume par une approche de type stimulus-réponse. En passant simplement par l'observation objective du comportement, le behaviorisme espère caractériser ce dernier pour pouvoir le contrôler. La méthode est donc en opposition avec la psychanalyse qui se concentre exclusivement sur l'introspection de faits mentaux difficilement mesurables.

Le système de base proposé par Watson est que le comportement observé d'une personne n'est qu'une réponse à un stimulus venu de l'environnement. Ainsi, deux personnes soumises au même stimulus auront les mêmes réponses comportementales, d'après cette théorie. De plus, une même personne soumise n fois au même stimulus aura n fois la même réponse.

Ce modèle comportemental a été critiqué car ne rendant pas compte de plusieurs phénomènes complexes. Skinner [Ski90] a proposé d'étendre le modèle sous la forme stimulus-réponse-conséquence. Ainsi, un comportement n'est plus seulement la réponse à un stimulus de l'environnement, c'est aussi une action prise en considération des conséquences possibles sur l'environnement. Des expériences menées en laboratoire montrent que les animaux sont capables d'apprendre de leurs erreurs, ce qui indique que c'est bien une projection des conséquences de leurs actes qui les animent, et non un stimulus perçu.

Pour autant, si ce nouveau modèle permettait de combler des manques dans la compréhension du comportement humain, la phase de raisonnement des sujets est toujours passée sous silence, comme placée dans une boite noire. C'est cette idée qui va être combattue par l'approche cognitiviste dans ce qui sera plus tard nommé la "révolution cognitive" [Gar93].

Cette révolution a eu lieu en 1956 avec une série d'articles montrant les limites du behaviorisme par des expériences en laboratoire. Miller [Mil56] montre notamment que l'être humain semble limité dans le nombre de concepts qu'il est capable de manipuler pour résoudre un problème dans un temps assez court. Par cette découverte, il est montré que la boite noire que représentent les processus de raisonnement d'une personne doit être étudiée car elle conditionne le comportement. Dans le même temps, Bruner [Bru17] publie un article sur les notions de catégorisation et de conceptualisation, deux phénomènes mentaux qui ne sont que du raisonnement et qui n'impliquent pas un comportement observable. Le behaviorisme ne peut donc pas les étudier alors qu'ils sont une part importante du raisonnement.

Le principe de la psychologie cognitive est d'étudier la cognition, c'est à dire l'ensemble des processus mentaux mis en jeu dans la fonction de connaissance. Cela rassemble les concepts de mémoire, de langage, de raisonnement, d'apprentissage, d'intelligence, de résolution de problème, de prise de décision, de perception ou encore d'attention [Gar93]. Chacun de ces concepts représente un champs important de travail, nécessitant la combinaison de différents domaines scientifiques pour être étudié. C'est ainsi que sont nées les sciences cognitives qui regroupent la psychologie cognitive avec la philosophie, la linguistique, l'anthropologie, les neurosciences et l'intelligence artificielle. Ces sciences cognitives ont aujourd'hui pour but d'étudier les systèmes complexes permettant de traiter de l'information afin de nourrir le raisonnement et la prise de décision.

2.1.2 Psychologie cognitive appliquée à l'IA

L'intelligence artificielle joue un rôle important sur la psychologie cognitive dès ses débuts avec l'allégorie du cerveau-ordinateur. Lors de son lancement en 1956, il est expliqué que le cerveau pourrait être vu comme un ordinateur qui traite des symboles acquis en entrée pour calculer un résultat en sortie. Le parallèle avec l'ordinateur se fait sur plusieurs points à savoir un dispositif d'entrée de l'information, la capacité de mémoriser de l'information symbolique, la possibilité de traiter cette information symbolique et enfin la capacité à exprimer le résultat de ce calcul. La cognition humaine ne serait alors qu'une suite de fonctions exécutées dans un certain ordre par le cerveau.

Cette vision de la cognition a été critiquée depuis, mais elle montre les liens importants qui existent entre psychologie cognitive et intelligence artificielle. Par la suite, une partie de l'IA s'est

servie des théories psychologiques pour développer des programmes de résolution de problèmes complexes plus proches du fonctionnement humain. Ces recherches ont menées à la création d'architectures comportementales comme discuté en section 1.3.

La première architecture importante à se baser sur la psychologie cognitive pour le comportement de ses entités informatiques est l'architecture SOAR (State Operator And Result) [LNR87], représentée dans une version récente par la figure 2.1. Le but de cette architecture était d'être la plus complète possible tout en étant le plus proche possible des théories en psychologie cognitive.

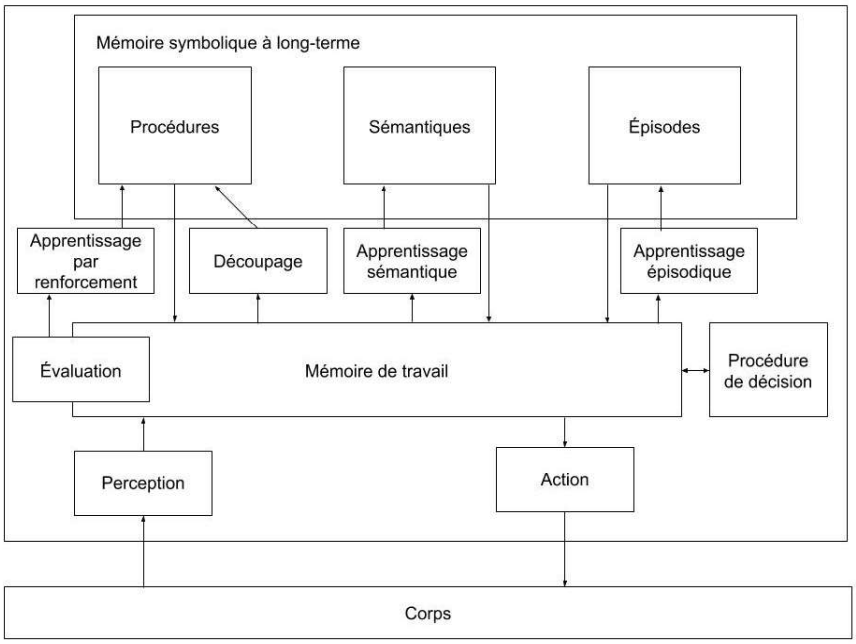

FIGURE 2.1 – Schéma de l'architecture SOAR, adaptée de [Lai12]

La version initiale de l'architecture n'avait pas d'apprentissage par renforcement ni de représentation sémantique ou épisodique dans sa mémoire à long-terme. Le déroulement de l'architecture consistait en une perception du problème, une mémoire à court-terme qui permet de faire le lien entre le processus de prise de décision et un ensemble de procédures contenues dans la mémoire à long-terme permettant de répondre au problème puis à l'activation effective de l'action sélectionnée. Une étape de découpage (chunking en anglais) permet de réduire un problème complexe à une somme de problèmes simples que la mémoire à long-terme peut mémoriser pour être plus efficace plus tard.

SOAR se veut générique et pour cela, l'architecture propose une représentation unique de tous les faits de connaissance, qu'ils soient dans la mémoire à court-terme ou à long terme. Par la suite, l'architecture a continué à évoluer, ajoutant une base sémantique et une base épisodique dans la mémoire à long-terme. La base sémantique permet de donner un sens différent à des faits égaux alors que la base épisodique contient la mémoire contextuelle des perceptions passées.

SOAR a été appliquée pour simuler le comportement de pilotes de l'armée de l'air [JTLR93]. Cette application montre que SOAR peut être utilisée dans des cas de coopération entre agents sur des problèmes complexes.

Dans la même optique de proposer des architectures comportementales pour des systèmes d'intelligence artificielle en s'appuyant sur les bases théoriques de la psychologie cognitive, d'autres travaux ont été proposés comme les architectures ACT-R [BA98] et CLARION [SMP01], qui sont présentées en section 1.3.1.

Le philosophe Bratman a de son coté travaillé la question du raisonnement sur les intentions à venir, donnant naissance au modèle BDI (Belief Desire Intention) [Bra87] en s'appuyant sur le concept de psychologie naïve qui impose d'expliquer les fonctionnements de la pensée humaine avec des concepts simples que l'on retrouve dans la vie quotidienne [Nor09].

Bratman ajoute les intentions aux traditionnels systèmes représentant les informations connues de l'agent soit comme des faits (les croyances), soit comme des volontés d'agir (les désirs). Les intentions sont alors vues comme une planification des actions à effectuer pour répondre à un désir particulier auquel la personne s'engage. Cela résout le problème de la délibération en continu des désirs, puisqu'une fois qu'une intention est choisie en fonction des désirs, l'agent va s'efforcer de la réaliser, tant qu'elle est encore réalisable (c.a.d tant que le contexte n'interdit pas de la réaliser).

2.1.3 Formalisations du modèle BDI

Le modèle BDI reposant par construction sur des concepts simples à manipuler, il s'est rapidement prêté à une adaptation informatique. La première étape vers une implémentation du modèle a été d'en proposer un formalisme logique ; Cohen et Levesque [CL90] une formalisation du modèle BDI en s'appuyant sur la logique modale. Les auteurs partent de quatre concept de base : Belief, Goal, Happens et Done, avec les définitions suivantes :

— **(BELIEF x α)** : l'agent x possède une croyance sur α.

— **(GOAL x α)** : l'agent x possède un but sur α.

— **(HAPPENS a)** : l'action a est réalisable à l'avenir.

— **(DONE a)** : l'action a vient d'être réalisée.

En partant de ces concepts et d'une définition logique rigoureuse, les auteurs définissent d'abord des propriétés assurant la cohérence au sein d'un agent utilisant ce système pour raisoner. Ainsi, il n'est pas possible de croire en même temps une chose et son contraire ou d'avoir un but sur une chose et son contraire.

En utilisant ces notions dans des formules logiques, Cohen et Levesque décrivent le concept de P-GOAL puis celui d'Intention, ce dernier étant le point central du modèle BDI. Ces notions sont définies de la façon suivante :

— **(P-GOAL x p)** : l'agent x possède un but persistant sur p. Cela signifie que x croit que p est faux et possède le but que p soit vrai dans le futur, jusqu'à ce que p soit vrai ou qu'il ne soit pus possible que p soit vrai.

— **(Intention x p)** : l'agent x possède une intention sur p. L'agent x possède un but persistant sur le fait que p soit réalisé s'il croit que p est une action réalisable à l'avenir.

Ce formalisme logique implique de nombreuses propriétés mais ne fonctionne que dans un monde "idéal". En pratique, le raisonnement de l'agent doit être moins rigoureux pour être calculable. C'est de le point de vue de Rao [RG+95] qui a proposé une formalisation du modèle BDI dans le but de l'implémenté dans une architecture comportementale, en s'appuyant sur les travaux de Cohen et Levesque.

Ainsi, Rao ne s'appuie que sur les notions Belief, Desire et Intention, qui symbolisent chacun des états du monde possible, rapport avec un monde des croyances, un monde des désirs et un monde des intentions. Si les concepts de croyances et de désirs sont repris de travaux précédents et sont proches des définitions de Cohen et Levesque, c'est le concept d'intention qui est le plus simplifié en le considérant de la même façons de que les croyances et les intentions (à savoir un état possible du monde qui n'est conservé que s'il est consistant et possible) tout en lui adjoignant la notion d'engagement.

De cette façon, une intention est non seulement une action que l'agent réalise, mais c'est surtout une action qui est réalisée tant qu'une condition de maintien de l'engagement est valide et qu'une condition de terminaison de l'engagement est fausse. Ces deux conditions permettent de ne pas reconsidérer l'action effectuée à chaque cycle de raisonnement. Ainsi, l'agent commence par mettre à jour ses croyances et désirs en fonction de son environnements, croyances et désirs qui peuvent être abandonnées à tout moment car n'étant pas soumis à la notion d'engagement. Il sélectionne ensuite une suite d'action atomiques à réalisée, en accord avec ses désirs et ses croyances, lui servant d'intentions. Une intention est sélectionnée et exécutée avant que l'agent

vérifie si certaines de ses intentions ont été satisfaites ou si certains de ses désirs ont été réalisés, mettant ainsi à jour ses bases de données.

Ce formalisme a été mis en œuvre à travers le langage de programmation AgentSpeak [Rao96], fondé sur les principe de la programmation logique de clauses de horn. Les croyances sont représentées sous la formes de prédicats avec un nom et une suite de termes (par exemple $b(t_1, ..., t_n)$), les buts sont des prédicats précédés du signe ! si ce sont des buts à atteindre et du signe ? si ce sont des buts pour vérifier une croyance. Enfin, des plans d'action sont définis avec une condition de déclenchement, un contexte à valider et une suite d'action. Les auteurs prennent l'exemple d'un robot devant ramasser des saletés puis les jeter à la poubelle. Un tel plan s'écrit comme l'indique l'équation (2.1) : $+\text{L}ocation(waste, x)$ indique que le plan est déclenché si l'agent acquiert la croyance que la saleté à ramasser se trouve à un lieu x, $\text{L}ocation(robot, x)\&\text{L}ocation(bin, y)$ indiquent les conditions nécessaires à ce que ce plan soit exécuté, $pick(waste)$; est une action atomique de prendre la saleté, $!\text{L}ocation(robot, y)$; permet d'ajouter le désir au robot de se déplacer vers l'endroit où se trouve la poubelle, une fois que ce désir aura été satisfait, le robot pourra réaliser l'action atomique $drop(waste)$;.

$$+\text{L}ocation(waste, x) : \text{L}ocation(robot, x)\&\text{L}ocation(bin, y) <- pick(waste); !\text{L}ocation(robot, y); drop(waste); \quad (2.1)$$

Ces formalismes ont conduit à la création d'architectures BDI pour le comportement d'agents, comme expliqué en section 1.2.3. Ces architectures sont devenues populaires, particulièrement dans la communauté des SMA et des simulations sociales [AG16]. Cette popularité s'explique par les racines dans la psychologie naïve du modèle BDI qui assure une simplicité de compréhension des mécanismes en jeu, tout en garantissant une grande profondeur des comportements possibles [Nor09] couplé à des formalismes logiques précis et pourtant calculables.

2.2 Ajout de caractéristiques affectives à la prise de décisions

La prise de décision d'acteur humains dans une situation particulière n'est pas exclusivement rationnelle. Elle inclut aussi des dimensions affectives comme les émotions ou la personnalité qui permettent d'expliquer pourquoi deux personnes, dans une même situation, ne prennent pas la même décision. Pour augmenter la crédibilité d'agents simulant des acteurs humains, il est donc nécessaire d'y inclure des dimensions affectives.

Jusqu'aux travaux de Damasio en 1994 [Dam94], les théories sur la cognition n'incluaient pas les dimensions affectives et sociales du comportement. Cela signifie que les émotions ou la personnalité n'étaient pas vu comme des facteurs à considérer dans les phénomènes cognitifs. Les travaux en IA qui se sont basés sur ces théories cognitives en ont donc adopté le vocabulaire, faisant une distinction entre des comportements qui seraient aujourd'hui qualifiés de rationnels face à des comportements affectifs ou sociaux.

À partir de 1994, les dimensions affectives et sociales ont été inclues dans la définition de la cognition, pour se mettre en accord avec les preuves expérimentales trouvées. Des expériences ont été menées avec des personnes ayant perdu une partie de leur cerveau à la suite d'accidents [Dam94], partie étant liée aux émotions d'après des rapports neurologiques. Lors de l'expérience, ces personnes n'ont pas réussi à prendre les mêmes décisions qu'un groupe test dans un jeu aux règles simples. L'étude a ensuite regardé des signes physiologiques pendant la même expérience, mettant en lumière le fait que la prise de décision, dans un jeu au premier abord purement rationnel, est quasiment uniquement lié à la capacité du sujet test à ressentir des émotions. Pour autant, la majorité des travaux menés en IA a prolongé la distinction entre les différents phénomènes, mettant en opposition d'un coté les comportements cognitifs et de l'autre les comportements émotionnels ou sociaux.

Cette distinction de vocabulaire et de sens est appliquée dans cette thèse. Ainsi, cette section s'intéresse aux dimensions affectives du comportement humain, et plus précisément aux émo-

tions et à la personnalité, en les considérant hors de la cognition qui ne représente, dans cette thèse, que les processus rationnels de la prise de décision. Le but de cette section est de montrer comment ces notions ont été étudiées en psychologie et quels résultats ont été utilisés en IA, et plus particulièrement en SMA et en simulation sociale.

2.2.1 L'apport des émotions dans la prise de décision

Les premières études scientifiques sur les émotions ont été menées en 1922 par James et Lange [LJ22]. Il a alors été conclu que les émotions étaient d'origine physiologique et se caractérisaient par la perception d'un changement physiologique induit par un stimulus. Ce n'est donc pas un phénomène qui fait intervenir le cerveau mais simplement un réflexe du corps à un événement de l'environnement. Par exemple, si quelqu'un perçoit une menace, il/elle va alors fuir. D'après James et Lange, c'est l'interprétation de cette fuite qui va induire une émotion, de peur dans ce cas.

Ce point de vue a été remis en cause par Cannon [Can27]. Du point de vue neurologique, les émotions semblaient se déclencher plus rapidement que la réponse physiologique. Depuis, ce point de vu a été en partie remis en cause, ouvrant la porte à de nombreuses propositions de théories émotionnelles. Ainsi, les émotions ont été vue prenant leur origine dans la motivation, le comportement, la communication ou encore la cognition.

Les théories émotionnelles utilisées en IA

Schachter et Singer [SS62] sont parmi les premiers à lier cognition et émotion. En se basant sur des théories sociologiques indiquant qu'une personne a toujours besoin de comprendre son état physiologique, il a été proposé que les émotions soient la manifestation de la compréhension d'un changement d'état physiologique à cause d'un stimulus. Ainsi, les émotions sont bien issu de la cognition, spécifiquement de la cognition sur un état physiologique.

En parallèle, Arnold [Arn60] développe la théorie de l'évaluation (appraisal en anglais) cognitive de l'environnement. Cette théorie propose que chaque personne évalue une situation perçue en fonction de critères personnels. Cela signifie qu'une même situation va être évaluée différemment en fonction de la personne qui l'évalue. Cette théorie implique alors que les émotions sont une réponse à cette évaluation cognitive de l'environnement, de même que la réponse physiologique créée. Pour reprendre un exemple, si une personne perçoit une situation menaçante, elle va évaluer cette situation comme étant une menace, ce qui va mener à l'apparition d'une émotion de peur en même temps que cela va indiquer à la personne de fuir.

Ce point de vu a été étendu pour donner naissance à la théorie de l'évaluation cognitive des émotions (cognitive appraisal theory of emotions en anglais) qui regroupe plusieurs théories reprenant ce principe d'émotions créées par la cognition [FKTS89] [SSE84] [JTLR93] [OCC90]. Si ces théories s'appuient toutes sur la même philosophie, elles diffèrent dans la description des mécanismes d'évaluation menant à l'apparition d'émotions. Pour autant, elles s'accordent toutes à définir une émotion comme la résultante de l'évaluation cognitive d'une situation.

En parallèle de cette théorie qui propose de définir un ensemble fini d'émotions, chacune liée à une règle de création, Russell et Mehrabian [RM77] ont proposé de représenter l'état émotionnel d'une personne comme un point dans un espace continu. Cette approche a été reprise en intelligence artificielle, en l'adaptant à un espace en deux dimensions [SG10] ou en trois dimensions [RCN+16] avec les dimensions plaisir, conscience et dominance, dans le cas d'un espace en trois dimensions. En fonction de la position dans cet espace, l'état émotionnel d'un agent exprime une émotion particulière. Cette théorie ne fournit pas de modèle concernant l'évolution temporelle de l'état émotionnel. Pour pallier à ce manque, Rincon [RCN+16] a proposé d'adapter la mécanique Newtonienne afin de ramener l'état émotionnel vers une position neutre, symbolisée par le centre de l'espace considéré.

La théorie de l'évaluation cognitive des émotions a été fortement adoptée dans la communauté des simulations à base d'agents [BTVA18], principalement parce qu'elle explique pourquoi

deux agents différents peuvent avoir des émotions différentes face à la même situation. De cette façon, l'hétérogénéité des agents est préservée et leur comportement peut évoluer dynamiquement en fonction de la situation perçue et des émotions ainsi créées.

Présentation détaillée de deux théories émotionnelles utilisées en SMA

Deux théories de l'évaluation cognitive des émotion ont principalement été utilisées en simulations sociales : celle proposée par Smith et Lazarus [SL+90] et celle proposée par Ortony, Clore et Collins [OCC90]. Elles sont toutes les deux développées dans cette section.

Smith et Lazarus partent du postulat qu'une personne évalue en continu son environnement. Les émotions sont alors créées en fonction des buts de l'agent et des ressources de l'environnement pour ensuite conditionner son comportement. Il y a donc deux phases distinctes : l'évaluation de l'environnement pour créer les émotions et l'application de stratégies pour faire face aux émotions créées.

La première phase consiste ainsi à déclencher des émotions en fonction de l'environnement perçu. Cette phase est composée par une première évaluation de l'impact de l'environnement sur le bien-être de l'agent, puis par une prise en compte des stratégies possibles pour faire face à la situation. Cela crée des schémas d'appréciations qui peuvent être définis à l'avance. Ainsi, des schémas d'évaluation cognitive sont proposés pour neuf émotions négatives (la colère, l'anxiété, la peur, la culpabilité, la honte, la tristesse, l'envie, la jalousie et le dégoût) et six émotions positives (la joie, la fierté, l'affection, le soulagement, l'espoir et la compassion). Pour prendre un exemple concret, la peur est défini comme "un danger soudain et concret d'une blessure physique imminente".

La seconde phase de la théorie contrôle la façon dont l'émotion créée agit sur l'action effectuée par l'agent, en prenant en compte un niveau de stress. En d'autres termes, ce processus consiste à minimiser le niveau de stress impliqué par l'émotion nouvellement créée. Des stratégies sont alors définies pour modifier l'action ou la cognition de l'agent parmi lesquelles sont proposées de trouver un soutient social, de se tourner vers la religion, de renier la réalité ou encore de reporter à plus tard les processus en cours afin de se concentrer sur la tâche causant l'émotion.

Cette théorie est très complète puisqu'elle décrit à la fois le mécanisme de création des émotions mais aussi leur impact sur la cognition. Pourtant, elle reste compliquée à implémenter car elle repose sur des notions subjectives comme les schémas d'évaluation cognitive qui ne sont pas formalisés en termes informatique.

Ortony, Clore et Collins ont développé une autre théorie fondée sur l'évaluation cognitive des émotions, connue sous le nom de théorie OCC, qui se concentre sur le processus de création des émotions. De leur point de vue, les émotions sont des réponses valuées à l'évaluation de trois types de stimulus : les événements, les actions réalisées par les agents et les objets. Cette évaluation est réalisée par rapport à trois variables principales : la désirabilité d'un événement en fonction des buts, la valeur morale des actions et l'apparence des objets. Chacune de ces variables principales permet de créer un embranchement, avec des variables secondaires pour aider à la catégorisation de l'émotion.

Le modèle OCC explique donc qu'une personne va évaluer une situation en fonction de ses trois variables principales pour détecter si l'émotion à créer porte sur un événement, sur une action ou sur un objet. Ensuite, l'évaluation est plus fine, en onction des variables secondaires, pour déterminer précisément l'émotion ressentie dans la situation courante.

Ainsi, vingt deux émotions sont définies, groupées en onze paires comme le montre la figure 2.2. Si ce modèle est devenu populaire, c'est surtout parce qu'il se prête assez simplement à une implémentation informatique. Chaque émotion est définie comme le point de sortie d'un arbre de décision, où chaque nœud correspond à un test sur une valeur numérique précise. Il y a donc un nombre fini de variables bien définies à prendre en compte. Par contre, ce modèle ne propose rien pour l'interaction entre les émotions et la cognition, une fois les émotions créées.

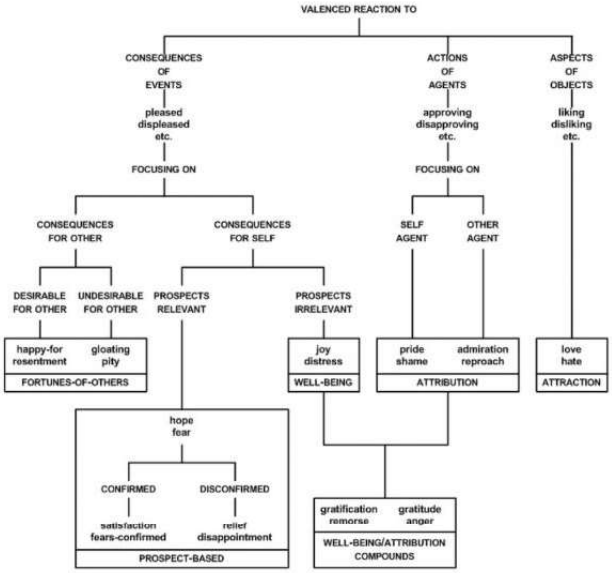

FIGURE 2.2 – Schéma du modèle OCC, tiré de [OCC90]

Représentations formelles des émotion

Ces différentes théories, issues d'études en psychologies, ont été formalisées pour être intégrées dans des systèmes informatiques. Le but de ces formalismes est de traduire des concepts théoriques abstraits en des objets manipulables et calculable sur ordinateur.

La théorie émotionnelle développée par Smith et Lazarus [SL$^+$90] a été formalisée et intégrée dans l'architecture émotionnelle EMA [GM04]. Les auteurs utilisent douze variables d'appréciation permettant d'émettre un jugement sur une situation. Chacune de ces variables est quantifiable ce qui signifie qu'un score est attribué à chaque situation. En fonction des valeurs dans ces variables, des émotions sont créées. Par exemple, si une situation a une désirabilité positive et une probabilité d'arriver inférieur à 1 (mais supérieur à 0), elle va créer chez l'agent de l'espoir. Si, par contre, sa probabilité d'arriver est égale à 1, c'est de la joie qui est créée. De cette façon, les six émotions définies dans l'architecture EMA sont définies par leur règle de création associée.

La seconde partie de la théorie de Smith et Lazarus, qui traite de l'impact des émotions sur le comportement, est lui aussi pris en compte dans l'architecture EMA. Pour ce faire, les auteurs définissent quatorze stratégie permettant de faire face aux émotions. Ces stratégies sont liées aux règles de création des émotions, de sorte que l'évaluation d'une situation soit différente une prochaine fois, ce qui implique qu'une même émotion ne créera pas la même émotion chez l'agent.

En ce qui concerne la théorie OCC des émotions, elle a été formalisée de façon logique à travers le modèle BDI [Ada07]. Le but principal de ce formalisme est de représenter les vingt-deux émotions du modèle OCC dans les termes du modèle BDI. Les opérateurs modaux de base de ce formalisme sont décrits de la façon suivante :

— $After_\alpha\phi$: ϕ est vrai après que l'action α a été exécutée.

— $Before_\alpha\phi$: ϕ est vrai avant que l'action α a été exécutée.

— $Bel_i\phi$: l'agent i croit que ϕ est vrai.

— $Prob_i\phi$: pour l'agent i, ϕ est plus probablement vrai que $\neg\phi$.

— $Idl_i\phi$: idéalement, ϕ est vrai pour l'agent i.

— $Des_i\phi$: l'agent i désire que ϕ soit vrai.

— $G\phi$: désormais, ϕ est vrai.

— $H\phi$: ϕ est toujours vrai.

En utilisant ces opérateurs, des définitions formelles sont données pour chaque émotion du modèle OCC. Par exemple, le modèle psychologique décrit la joie comme l'émotion ressentie quand l'agent est satisfait à propos d'un de ses désirs, ce qui se traduit en logique formelle par l'équation (2.2)

$$Joie_i\phi \stackrel{\text{def}}{=} Bel_i\phi \wedge Des_i\phi \tag{2.2}$$

Sur le même modèle, une formalisation est proposée pour chaque émotion décrite par le modèle OCC, en traduisant la définition psychologique de chaque émotion par une formule logique en utilisant les opérateurs modaux du modèle BDI. Ce formalisme logique permet alors d'inclure des émotions dans un système informatique qui va pouvoir raisonner sur des règles précises.

2.2.2 Différencier les comportements avec la personnalité

La personnalité fait partie des facteurs de différenciation entre les personnes et plusieurs travaux ont montré que cette notion avait un impact sur le comportement de chacun [EE+87] [WC92], mais aussi sur les émotions [Ort02] ou encore sur la capacité à obéir aux normes [BBC+15]. Ainsi, la personnalité est vue comme un outil permettant de caractériser, a priori, différents comportements.

Les recherches en psychologie sur le sujet de la personnalité ont été majoritairement centrées sur la différenciation des multiples facteurs pouvant déterminer un comportement [Eys50]. Cattell [Cat66] propose le premier modèle émotionnel accompagné de la première étude d'envergure sur le sujet. La méthode utilisée a par la suite été reprise dans tous les travaux menées sur la personnalité à savoir une étude statistique de réponses proposées par des sujets de test à un questionnaire particulier. Le résultat a produit un modèle de personnalité à seize traits. Cela signifie que seize composantes, jugées à ce moment indépendantes, ont été ressorties de l'étude des questionnaires, permettant de catégoriser les différents comportements.

Chaque trait représente une échelle de valeurs couvrant l'ensemble d'un aspect de la personnalité. En d'autres termes, le trait d'extraversion est une échelle allant de la timidité à l'extraversion. Il est attribué un score à chaque individus pour chacun des traits et l'ensemble de ces scores représente la personnalité globale d'une personne.

Les résultats obtenus par Cattell ont été critiqués, notamment parce que certaines des notions semblaient plus liées au mot qui les décrivaient qu'à un réel concept. Ainsi, en traduisant les résultats ou en reproduisant l'expérience dans des pays de langues différentes, il était difficile de retomber sur le même modèle [Eys91]. Pour autant, la méthode employée est restée la même à savoir s'appuyer sur un questionnaire bien établi [E+64] puis analyser les réponses proposées d'un point de vue statistique pour en retirer un nombre minimal de traits mais suffisant pour décrire l'ensemble des comportements. Parmi les travaux importants proposées, il est à noter le modèle MBTI [MMM85] à quatre dimensions et le modèle PEN [Eys90] à 3 dimensions.

Le modèle MBTI propose de représenter la personnalité d'une personne selon quatre traits représentant des préférences :

— Orientation de l'énergie : indique si la personne tend à diriger son énergie vers les autres ou vers soi. Les deux modalités possibles sont Extraversion (E) et Introversion (I).

— Récupération de l'information : indique la façon privilégiée de récupérer de l'information pour résoudre un problème. Soit cette récupération se fait via les sens, soit elle se fait par réflexion. Les deux modalités possibles sont donc Sensation (S) et Intuition (N).

— Prise de décision : indique si la personne prend plutôt des décisions sur un raisonnement rationnel ou en laissant s'exprimer ses émotions. Les deux modalités possibles sont Pensée (T) et Sentiment (F)

— Mode d'action : indique la façon d'interagir avec le monde. Cette interaction peut être un processus actif ou un processus passif. Cela conduit aux deux modalités possibles qui sont Jugement (J) et Perception (P).

Puisqu'il y a à chaque fois deux modalités par trait de personnalité, le modèle MBTI définit seize personnalités différentes sous le format de quatre lettres représentant les modalités préférentielles d'une personne pour chaque trait. Par exemple, quelqu'un de tourné vers les autres, prenant ses informations de ce qui est ressenti, appliquant un raisonnement rationnel et agissant activement sur l'environnement aura une personnalité ESTJ.

Le modèle PEN s'appuie plus sur les travaux de Cattell que le modèle MBTI, en essayant de réduire au maximum le nombre de traits nécessaires pour représenter une personnalité. Les mêmes questionnaires que Cattell ont été utilisés et la même méthode d'étude statistique a été mise en œuvre pour aboutir à trois traits :

— **Psychotisme** : représente la propension d'une personne à garder son calme. Peut s'apparenter à une mesure du niveau d'agressivité d'une personne.

— **Extraversion** : indique la capacité d'une personne à s'exprimer et à se mettre en avant. Une personne avec une faible valeur d'extraversion est considérée comme timide.

— **Neurotisme** : représente la capacité d'une personne à contrôler ses émotions. Une forte valeur représente une personne qui se laisse submerger par son état émotionnel.

La personnalité de quelqu'un selon le modèle PEN est donc un ensemble de trois valeurs. Contrairement au modèle MBTI qui propose un nombre fini de personnalités, le modèle PEN représente la personnalité d'une personne comme un point dans un espace en trois dimensions continues.

Le modèle qui s'est finalement imposé, selon Eysenck [Eys91], a repris des éléments de PEN et de MBTI en repartant de la méthodologie de Cattell pour proposer cinq facteurs [MJ92] sous le nom de FFM (Five Factor Model) puis sous le nom OCEAN, en lien avec le nom des différents traits :

— **Ouverture** : représente la capacité d'ouverture d'esprit d'une personne. Une faible valeur correspond à quelqu'un qui a du mal à accepter de nouveaux faits et à se remettre en question alors qu'une forte valeur correspond à quelqu'un ouvert d'esprit, original et curieux.

— **Conscience** : représente la capacité d'une personne à agir avec préparation. Une faible valeur correspond à quelqu'un qui agit de façon impulsive, apte à improviser, alors qu'une forte valeur correspond à quelqu'un d'ordonné, qui prépare à l'avance ses actions.

— **Extraversion** : représente la capacité de quelqu'un à s'exprimer vers les autres. Une faible valeur correspond à quelqu'un de timide, réservé, alors qu'une forte valeur correspond à quelqu'un qui n'a pas peur de s'exprimer en public.

— **Agréabilité** : représente la capacité d'une personne à évoluer avec les autres. Une faible valeur correspond à quelqu'un d'égocentré, mal aimable avec les autres, alors qu'une forte valeur correspond à quelqu'un tourné vers l'aide aux autres, quelqu'un d'amical.

— **Neurotisme** : représente la capacité d'une personne à contrôler son état émotionnel. Une faible valeur correspond à quelqu'un capable de surmonter ses émotions alors qu'une forte valeur correspond à quelqu'un qui se laisse dépasser par ses émotions.

Ce modèle à cinq facteurs s'est imposé comme le modèle principal de personnalité parce qu'il est celui qui fait le moins débat parmi les psychologues [Eys91]. Chaque trait est assez vague pour y inclure de nombreux comportements hétérogènes tout en étant bien définit les uns par rapports aux autres. Ce modèle a notamment été utilisé en simulation pour connecter plusieurs dimensions sociales dans la définition de comportements agents [LLB12] [OSC09b]. Dans ces deux cas, la personnalité d'un agent est représentée par un vecteur à cinq dimensions, où chaque valeur, représentant une composante du modèle OCEAN, est un nombre réel entre -1 et 1.

2.3 Prise en compte du contexte social dans la prise de décision

Le domaine des simulations sociale s'intéresse non seulement à des agents simulant des acteurs humains, mais surtout à des groupes d'agents, évoluant dans une même situation et interagissant entre eux. Cette interaction sociale a un impact doit être prise en compte dans la prise de décision de tels agents.

Après les travaux de Damasio [Dam94], il est apparu important d'intégrer dans le concept de cognition des dimensions qui en avaient été exclues jusque là. C'est ainsi que les dimensions affectives et sociales ont été considérées comme des pans importants de la cognition. Au contraire des dimensions affectives qui ont été incluses directement dans le concept de cognition, un nouveau champs, la cognition sociale, a été créé pour associer les processus du raisonnement à la prise en compte du contexte social.

Un processus social est un comportement centré autour de la relation avec les autres personnes en contact. Contrairement aux dimensions cognitives et affectives, qui s'intéressent aux comportements de quelqu'un en fonction de l'environnement, la dimension sociale ne prend en compte que les comportements vis à vis des autres personnes possédant eux aussi une dimension sociale. C'est donc un processus important dans les simulations sociales qui visent à reproduire le comportement d'acteurs humains pris dans des situations où ils sont en contact avec d'autres acteurs humains et doivent donc adapter leur comportement par rapport aux autres.

Dans cette partie il est notamment question de l'évolution des émotions dans un contexte social, de la prise en compte des relations interpersonnelles ainsi que du comportement dans une société normée. Ces notions sont par essence liées à la nature sociale de l'humain et ont un impact sur la prise de décision individuelle.

2.3.1 Évolution des émotions dans un contexte social

Dans un contexte social, les émotions se propagent dans un réseaux, faisant évoluer l'état émotionnel de chaque personne de ce réseaux. C'est notamment comme cela que se créent les fous rires, où la joie d'une personne se propage parmi les gens percevant cette joie. Ce phénomène est étudié sous le nom de contagion émotionnelle [HCR93] et est défini par Hatfield, dans un premier temps, comme "la tendance à automatiquement mimer et synchroniser les expressions faciales, la posture, les intonations de la voix et les mouvement d'une autre personne perçue et ainsi de converger émotionnellement vers cette personne."

Dans son étude, Hatfield explique la contagion émotionnelle par plusieurs observations sur des personnes qui tendent à homogénéiser leurs comportements et notamment leurs expressions faciales, avec les gens autour d'eux. Les expériences montrent qu'une personne, mise en face de photographies d'acteurs jouant des expressions de joie ou de tristesse, modifie de façon quasi-instantanée son expression faciale pour correspondre à celle de l'image. D'autres études ont été menées par la suite, montrant que les émotions peuvent se propager à travers un réseau social en ligne [KGH14], soulignant ainsi que ce processus est lié aux émotions véhiculées d'une personne à une autre, que ces personnes soient en contact physique ou non.

Barsade [Bar02] a mené une étude sur le même sujet avec des groupes de parole. Un acteur était dissimulé au milieu des sujets de tests, jouant soit une émotion de joie, soit une émotion de tristesse. Il est alors ressorti que le présence de cet acteur modifiait l'état émotionnel de chacune des personne testée, mais aussi que les sujets tests, entre eux, se contaminaient par la suite, jusqu'à une certaine limite. L'étude a aussi conduit ses auteurs à élargir la contagion émotionnelle à la création d'émotions en fonction des émotions des autres. Par exemple, lorsque quelqu'un est humilié en public, la joie des autres va conduire à augmenter la tristesse de la personne humiliée. La contagion n'est donc pas seulement un processus de copie des émotions des autres.

Ainsi, la contagion émotionnelle peut être décrite plus largement comme le processus par lequel un agent adapte ses émotions en fonction des émotions des personnes qui l'entourent, que ce soit physiquement ou via un réseau social. Ce processus est rapide et s'appuie sur l'analyse cognitive de chacun pour, soit copier la moyenne des émotions perçues, faisant converger l'état

émotionnel du groupe, soit créer une émotion en rupture avec le reste du groupe.

Cette définition a été formalisée par le modèle ASCRIBE (Agent-based Social Contagion Regarding Intentions Beliefs and Emotions) [BDM+09] qui propose de modéliser le processus de contagion émotionnelle entre deux agents en se basant sur les cinq variables suivantes :

— q_E : intensité de l'émotion transmise par l'agent émetteur E.
— q_R : intensité de l'émotion ressentie par l'agent receveur R.
— μ_E : force avec laquelle l'agent E transmet son émotion.
— δ_R : capacité de l'agent R à être contaminé.
— α_{ER} : force du canal d'échange entre E et R.

Le modèle définit aussi une formule de calcul pour l'évolution de l'émotion du receveur selon l'équation différentielle (2.3) avec G l'ensemble des agents perçus et q_R^* la moyenne de l'intensité de l'émotion de chaque agent émetteur appartenant à G, pondéré par μ_E et α_{ER}.

$$\frac{dq_R}{dt} = \sum_{E \in G} (\mu_E \times \alpha_{ER} \times \delta_R)(q_R^* - q_R) \tag{2.3}$$

Enfin, il convient de distinguer contagion émotionnelle et empathie. L'empathie est un processus qui consiste, pour une personne, à se mettre émotionnellement à la place d'une autre personne [Koh66]. Dans ce processus, le personne ressentant de l'empathie ne modifie pas son propre statut émotionnel; il y a une dissociation qui s'effectue, permettant à la personne de ponctuellement ressentir les émotions d'une autre personne, sans pour autant changer ses propres émotions. Pour autant, il est possible de considérer que l'empathie crée de la contagion émotionnelle, au sein d'une même personne. La contagion a lieu entre la personne qui ressent de l'empathie et la partie dissociée de cette même personne qui se met à la place de la personne perçue.

2.3.2 Des relations sociales pour un comportement en société

Bien qu'il n'existe aucune étude précise montrant l'impact des relations sociales sur le comportement de chacun, il parait évident que les gens ne se comportent pas de la même façon en présence de parfaits inconnus ou entourés de personnes déjà rencontrés. C'est pour cette raison que les relations sociales ont été étudiées, avec comme défi principal leur représentation.

Deux approches se font face sur la question de la façon de considérer les relations sociales. La première façon de considérer les relations sociales est par le prisme de rôles institutionnels [RS62]. Cela correspond aux relations de type professeur-étudiant ou policier-hors la loi; un rôle social est donnée à une personne dans un cadre précis. L'interaction entre deux personnes va alors se faire en fonction du rôle de chacun dans ce cadre précis. Prenons l'exemple de deux personnes, A et B, exerçant le métier d'agent de police. Pour autant, l'agent B a commis une infraction pour laquelle l'agent A le poursuit. Dans ce cadre, l'agent A sera considéré comme un policier et l'agent B comme un hors la loi dans leur relation sociale ce qui affectera la position et le comportement de chacun.

Cette vision des relations sociales impose d'avoir un cadre social bien établi avec des rôles précis partagés par tous les agents du système. Cela impose aussi de considérer tous les échanges possibles a priori à l'intérieur de ce cadre, ce qui est assez contraignant. De plus, cela ne permet pas de capturer tous les types de relations sociales avec assez de finesse. Prenons l'exemple d'une famille au sens large. Dans cette famille, des personnes peuvent avoir le même rôle social (il peut y avoir plusieurs oncles ou tantes par exemples) mais ne pas être traitées de la même façon (certains oncles ou tantes seront plus appréciés que d'autres).

L'autre façon de considérer les relations sociales est sous l'angle des relations interpersonnelles, ce qui a été fait par Rousseau [RHR98], Walker [Gra02] ou encore Gratch [Gra02]. Dans l'ensemble des ces travaux, les relations sociales entre les agents sont représentées par un nombre fini de variables numériques qui correspondent aux variables minimales permettant de décrire une relation entre deux personnes, données dans le modèle dimensionnel des relations interpersonnelles proposé par Svennevig [Sve00] :

— **Appréciation** : indique si la personne vers qui est dirigée la relation est appréciée ou détestée [Isb06] [PI01].

— **Dominance** : indique si la personne vers qui est dirigée la relation domine la relation ou est dominée par elle [RHR98] [PI01].

— **Solidarité** : aussi appelée distance sociale [BL87] [WCW97], indique le degré de similarités d'états mentaux, en terme de buts, de désirs et de croyances, avec la personne vers qui est dirigée la relation [BC01].

— **Familiarité** : indique la qualité et la quantité d'informations possiblement échangées avec la personne vers qui est dirigée la relation [BC01].

Ainsi, une relation sociale entre deux personnes est décrite par un vecteur de valeurs numériques qui n'a pas besoin d'un cadre social pour être significatif. Cette relation peut aussi être créée entre deux personnes, sans avoir à les identifier dans un rôle précis. De plus, la description des relations est ici plus précise qu'avec l'approche en rôles puisque de très petites différences, en termes numérique, peuvent permettre de classer des relations sociales entre elles.

Il convient aussi de noter que les relations interpersonnelles par variables ne sont pas nécessairement symétriques entre elles. Si l'ensemble des quatre variables s'exprime entre 0 et 1, un agent A peut avoir une relation sociale envers B avec les valeurs (0,0,0,0) alors que B peut avoir une relation sociale envers A avec les valeurs (1,1,1,1). Cela signifiera simplement que les deux agents n'ont pas la même évaluation de leur relation, l'un des agents détestant l'autre quand cet autre apprécie fortement le premier.

Les deux approches ne sont pas pour autant obligatoirement opposées. Ochs [OSC09b] propose notamment de représenter les relations sociales d'agents simulant des humains avec le modèle dimensionnel des relations interpersonnelles mais se sert d'une représentation en rôle pour initialiser ces relations sociales. Cela signifie qu'il est fait une correspondance entre le fait de faire face à un policier quand on est un hors la loi et les valeurs qui sont données à la relation sociale avec ce policier. La personnalité de chacun est intégrée pour différencier l'initialisation de ces relations entre plusieurs agents. Ainsi, les relations sociales ont une description fine mais garde un sens dans le contexte social dans lequel elles sont utilisées.

Pour la question de la création des relations sociales, il y a là aussi deux façons de voir les choses. L'approche statique [RS62] consiste a indiquer qu'une relation sociale existe par le biais d'un cadre social bien défini. Cela signifie que la relation sociale existe a priori, avant même que les personnes ne se rencontrent. Par exemple, supposons que B soit l'oncle de A. Avant même que ces personnes ne se rencontrent ou n'aient connaissance l'un de l'autre, ce lien social existe et s'activera une fois qu'elles se seront rencontrées.

L'autre approche est une vision dynamique de la création des relations sociales [Zaj65]. Des relations ne se créent qu'au moment d'une prise de contact avec une autre personne, inconnue jusque là. Ce lien alors créé va perdurer dans le temps, sans qu'il y ait besoin de de nouveau être en contact avec la personne concernée. Zajonc [Zaj65] explique que des expériences ont été menées dans ce sens, en mettant artificiellement en contact des personnes inconnues pendant une soirée dansante. Les sujets de test étaient alors capables, plusieurs semaines plus tard, de caractériser leur relation avec les gens rencontrés ce soir là, sans s'être revu depuis.

Encore une fois, ces deux approches peuvent êtres complémentaires, en fonction de la situation étudiée. Il est possible d'imaginer une situation où une famille se déplace et interagit avec des inconnus. Les membres de la familles auront alors les uns pour les autres des liens créés a priori par leur appartenance à la construction sociale qu'est la famille, mais ils créeront aussi chacun des relations sociales dynamiques avec les inconnus rencontrés.

Pour la question de l'évolution des relations sociales dans le temps, plusieurs travaux ont montré que cette évolution est liée aux autres processus cognitifs d'une personne, et en particulier à ses émotions [OKC91] [dRG86] [PF04]. Le seul essai, à notre connaissance, d'évolution des relations sociales entre agents en se basant sur les émotions est proposé par Ochs [OSC09b] qui définit manuellement un ensemble d'émotions agissant positivement ou négativement sur les différentes

dimensions du modèle dimensionnel de relations interpersonnelles. La figure 2.3 montre l'ensemble des émotions influençant les dimension d'appréciation et de dominance d'une relation sociale entre deux agent. Ces relations sociales évoluant dynamiquement sont ensuite utilisées pour modifier la prise de décision de l'agent.

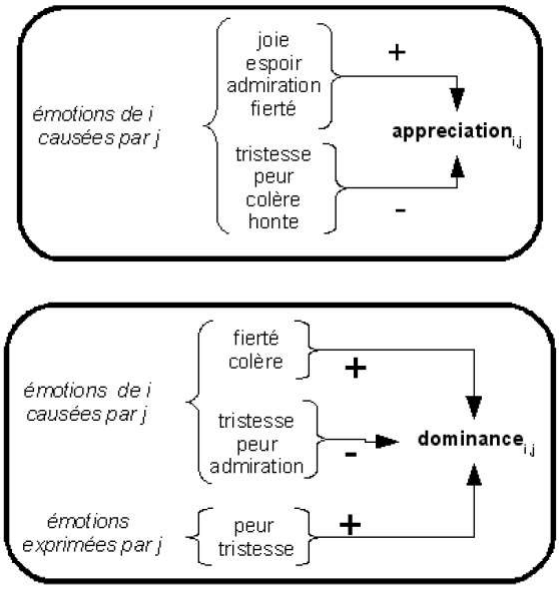

FIGURE 2.3 – Schémas de l'évolution des relations sociales en fonction des émotions, tiré de [OSC09a], p.19

2.3.3 Agir dans une société normée

Introduction aux systèmes normatifs

Les normes permettent de décrire le comportement attendu d'un système, qu'il soit social ou non. Un ensemble de normes indique à un système, au niveau macroscopique, comment celui-ci doit se comporter, ainsi que ce qui y est possible et ce qui y est interdit.

Gibbs [Gib65] a uniformisé les définitions existantes du concept de norme en considérant une norme comme un processus impliquant une évaluation collective des comportements par rapport à ce qui doit être fait, par rapport à une attente collective des comportements à venir ainsi que par rapport à une réaction face aux comportements observés, cette réaction pouvant être une sanction. Cette définition assez large permet de prendre en compte aussi bien les lois explicites imposées par une institution et qui doivent être suivies sous peine de sanctions matérielles que les normes sociales implicites partagées par une société et qui peuvent être soumises à des sanctions sociales.

Tuomela [Tuo95] propose de distinguer quatre classes de normes :

— Les R-normes (Rule Norm) : des normes strictes et explicites imposées par une institution et qui ne peuvent être ignorées. Cela représente les lois d'un état, d'une entreprise, etc.

— Les S-normes (Social Norm) : des normes implicites qui représentent une convention sur un comportement normal à adopter dans une société. Cette norme sociale est partagée implicitement par plusieurs membres d'une même société et n'est valable qu'auprès de ces membres, qui sont responsable de son contrôle.

— Les M-normes (Moral Norm) : les normes qui sont imposées par soi-même et qui ne s'appliquent qu'à soi-même. Par exemple, le fait de suivre un précepte religieux dans la sphère privée, là où personne ne peut contrôler l'application de la norme.

— Les P-normes (Prudential Norm) : les normes qui assurent le fonctionnement de base de maximisation de l'utilité des actions. Ces normes indiquent par exemple qu'il faut manger pour continuer à vivre.

Tuomela définit un ordre dans lequel les R-normes sont prioritaires sur toutes les autres normes. De plus, le type des normes peut évoluer dans le temps. Ainsi, si une M-norme est acceptée par un assez grand nombre de personnes, elle devient une S-norme qui, si elle est aussi acceptée par un très grand nombre de personnes, alors l'institution au dessus de ces personnes peut décider de l'expliciter, la transformant en R-norme. Enfin, à chaque type de norme correspond un type de sanction :

— Les R-sanctions : appliquées par une institution, elles sont connues de tous et explicites avant leur application. Elles viennent sanctionner la violation d'une R-norme.

— Les S-sanctions : appliquées par les individus partageant la S-norme violée, elles sont implicites et peuvent modifier des valeurs sociales partagées avec la personne sanctionnée.

— Les M-sanctions : appliquées par l'individu lui-même en réponse à la violation d'une M-norme, elles sont implicites et uniquement connues de l'individu qui se l'inflige lui-même.

— Les P-sanctions : appliquées par le système dans lequel évolue l'individu violant la P-norme correspondante, elles sont implicites et vont modifier le comportement de base de l'individu.

Une autre catégorisation est proposée par Therborn [The02] qui s'appuie sur trois classes de normes :

— Les normes constitutives : indiquent les actions possibles ainsi que les individus sur lesquels s'applique le système normatif.

— Les normes régulatrices : définissent les apports attendus de l'individu au système social.

— Les normes distributives : définissent les coûts, les récompenses et les risques inclus dans un système social.

Par dessus cette catégorisation, Therborn indique que le système normatif doit indiquer quelles sont les normes institutionnelles, c'est à dire les normes qui s'appliquent à des agents ayant des rôles particuliers. Cette notion permet, entre autre, de traiter la question de l'évolution dynamique des normes. Ainsi, si un agent entre dans une institution, il obtient un rôle ce qui lui indique aussitôt les normes qu'il doit suivre. Pour les agents sans rôle, c'est l'observation du comportement des autres agents qui va permettre de se représenter les normes, de les accepter et de les suivre.

Savarimuthu [SC09] s'est intéressé au cycle de vie vie des normes et le décrit en quatre phases :

— Création : les normes doivent d'abord être définies puis transmises à au moins un agent du système.

— Propagation : une fois introduites dans le système, les normes doivent être propagées. Pour cela, différents mécanismes peuvent entrer en jeu, comme le leadership, l'imitation ou le pouvoir social.

— Contrôle : pour renforcer leur acceptation et leur importance, l'application des normes doit être contrôlée. Ce contrôle est effectué par des agents du systèmes qui appliquent les sanctions prévues en cas de violation de norme.

— Émergence : finalement, des normes peuvent émerger du système si un grand nombre d'agents effectue le même comportement. Ce comportement adopté par tous devient alors une nouvelle norme.

Ce cycle de vie implique que le système normatif peut évoluer au cours du temps, en fonction des actions des agents, possiblement dans des directions non prévues au départ.

Formalismes des systèmes normatifs

Finalement, le concept de norme a été formalisé pour être inclus dans des systèmes informatiques. Stratulat [Str02] propose une formalisation de tous les types de normes possibles en s'appuyant sur la logique déontique. Le but est de permettre à un agent de raisonner de façon logique sur l'ensemble des normes qu'il possède, que ce soit des obligations, des permissions ou des interdictions. Pour cela, trois prédicats déontiques sont déclarés :

— O(agent,α,i) : indique qu'un $agent$ est obligé de réaliser l'action α dans l'intervalle de temps i.

— P(agent,α,i) : indique qu'il est permis à un $agent$ de réaliser l'action α dans l'intervalle de temps i.

— I(agent,α,i) : indique qu'il est interdit à un $agent$ de réaliser l'action α dans l'intervalle de temps i.

Le système normatif se résume alors à une suite de formules sous la forme $\phi \rightarrow$ OPI signifiant que si ϕ est vrai, cela implique l'un des prédicats déontiques présenté ci-dessus. Chaque prédicat est ensuite valide pendant un certain temps au cours duquel il peut être violé.

Pour Dignum [Dig99], un système normatif se compose de trois couches : le niveau privé, le niveau contractuel et le niveau conventionnel. Le niveau privé permet de définir ce qu'un agent préfère pour lui-même, ce qui représente une norme qu'il s'impose. Le niveau contractuel définit les échanges dynamiques entre les agents. Enfin, le niveau conventionnel définit le limites imposées par le système aux agents, en terme d'interdictions et de permissions. Il est donc nécessaire de définir les prédicats déontiques suivants :

— $Pref_i(\phi|\psi)$: l'agent i préfère ϕ à ψ chaque fois que ϕ et ψ sont simultanément possibles.

— $O_{i,j}(\alpha)$: l'agent i s'engage par obligation envers l'agent j pour réaliser α.

— $auth(i,\alpha)$: l'agent i est autorisé, par un autre agent quelconque à réaliser α.

— $F_i(\alpha)$: il est interdit, par le système, à l'agent i de réaliser α.

— $P_i(\alpha)$: il est permis, par le système, à l'agent i de réaliser α.

Toujours en utilisant l'approche de la logique déontique, Hübner [HSB02] propose de lier les spécifications structurelle d'un système à sa spécification fonctionnelle en utilisant les concepts d'obligation et de permission. La spécification structurelle permet de décrire statiquement un système, en termes de rôles, de liens entre ces rôles et de groupes, indiquant la structure globale dans laquelle évoluent les agent. En parallèle, la spécification fonctionnelle décrit le fonctionnement du système sous la forme de missions de haut niveau qui se décomposent en un suite de sous-plans. Afin de lier ces deux types de spécification, les auteurs définissent les deux prédicats déontiques suivants :

— $obl(\rho, m, tc)$: un agent ayant le rôle ρ est obligé de réalisé la mission m sous la contrainte de temps tc.

— $per(\rho, m, tc)$: il est permis, à un agent ayant le rôle ρ, de réalisé la mission m sous la contrainte de temps tc.

La définition de plusieurs de ces prédicats déontiques permet alors de lier la représentation structurelle du système à sa représentation fonctionnelle, et ainsi d'assurer son fonctionnement dans un contexte normatif.

Une autre approche est proposée par Lopez y Lopez [yLLd06] qui s'intéresse plutôt à l'unification de la représentation des différents concepts nécessaires à un système normatif. La proposition consiste à tout considérer comme une norme, que ce soit une obligation, une norme sociale, une sanction ou une récompense. Dans ce cas, une norme est composée des éléments suivants :

— Un but à satisfaire, indiquant ce que l'agent doit faire.

— Un ensemble d'agents pour lesquels cette norme s'applique, indiquant les agents devant respecter cette norme.

— Un ensemble d'agents vers qui la norme est dirigée. Cet ensemble peut être vide si la norme n'a pas d'implications sur les autres agents.
— Un contexte dans lequel la norme est activée.
— Un contexte d'exception dans lequel il est possible de violer la norme sans risquer de sanctions.
— Une récompense indiquant ce que l'agent obtient s'il se conforme à la norme.
— Une sanction indiquant ce que l'agent subit s'il viole la norme.

Ainsi, du point de vue de Lopez y Lopez, les sanctions, récompenses, contextes et contextes d'exceptions peuvent s'exprimer comme des normes, selon le formalisme proposé ci-dessus. De cette façon, le système n'est qu'un enchaînement de normes qui peut alors se voir sous la forme d'un arbre des comportements possibles.

2.4 Conclusion

Ce chapitre montre que la question de la prise de décision, chez l'humain, ne se résout pas simplement à un comportement réflexe qui se résumerait à un ensemble de règles simples du type "si-alors"sinon". Le raisonnement rationnel seul implique d'inclure des concepts de mémoire, d'apprentissage et de communication comme l'explique la psychologie cognitive [Gar93]. Il convient ensuite d'ajouter des processus affectifs et sociaux permettant d'expliquer les comportements irrationnels humains pour se rapprocher d'un modèle réaliste de la prise de décision d'une personne.

De plus, ce chapitre montre, pour chaque notion psychologique présentée, que différents points de vue se font face dans la communauté des sciences cognitives. Il n'y a pas de consensus véritable sur les notions de cognition, d'émotions, de personnalité, de relations sociales, de contagion émotionnelles ou de normes. Ce sont aussi des sujets d'étude en cours dont la compréhension peut être partiellement remise en cause avec des recherches en neurologie, comme ce fut le cas lorsque Damasio [Dam94] remit en cause la conception d'une cognition détachée des émotions.

Il n'y a pas non plus de consensus sur les liens précis entre ces relations et le processus de décision. Il n'existe aucun modèle universel de la prise de décision indiquant clairement le rôle joué par chaque dimension psychologique. Pour autant, l'approche qui semble être prise par la communauté des sciences cognitives est de considérer la cognition rationnelle comme le processus central de la prise de décision, puis d'utiliser les processus affectifs et sociaux comme des filtres, modifiant l'information en entrée ou en sortie du moteur de raisonnement comme le montre les différentes approches comportementales proposées en IA et détaillées en section 1.3.

Pour simuler le comportement d'agents représentant des acteurs humains de façon crédible, nous proposons de nous inspirer des recherches détaillées dans ce chapitre. Il convient alors de choisir, pour chaque dimension du comportement, quelle théorie et quel formalisme inclure dans l'architecture développée dans cette thèse.

Cognition

Les sciences cognitives sont aujourd'hui au centre de la question de la prise de décision et elles ont amenées à la définition de plusieurs architectures cognitives.

D'un coté, les architectures SOAR [LNR87], ACT-R [BA98] et Clarion [SMP01] proposent des approches complexes, s'appuyant sur une différenciation entre mémoire à court terme et mémoire à long terme ainsi que sur une différenciation entre représentation explicite et représentation implicite de l'information. Ces architectures s'inspirent du fonctionnement bas niveau du cerveau, comme un miroir de l'approche cerveau-ordinateur utilisée pour définir les sciences cognitives.

De l'autre coté se trouvent les architectures BDI fondées sur une vision haut niveau du principe de cognition, amenant les agents à manipuler des états mentaux bien définis. Ces architectures sont devenues les plus populaires pour créer des agents sociaux cognitifs [AG16], principalement

parce que des implémentations sont déjà proposées dans les plateformes de simulation [TBC+16] [SKS08] mais aussi parce qu'elles permettent de complexifier le comportement des agents tout en restant à un niveau explicable.

L'architecture développée dans cette thèse va ainsi s'appuyer sur une architecture BDI pour assurer la cognition des agents. En plus d'être populaire auprès de la communauté des simulations sociales, le modèle BDI est issu de la psychologie naïve qui lui permet de servir de fondation à des systèmes incluant d'autres dimensions comportementales [Nor09].

Émotions

Il existe deux façons admises de représenter et de traiter les émotions en psychologie. La première de ces façons est de considérer l'état émotionnel d'une personne comme un point dans un espace continu [RM77]. Cette approche se prête bien à une implémentation informatique puisqu'elle ne consiste qu'en des calculs simples sur un vecteur en trois dimensions. Pour autant, cette théorie ne dit rien sur l'évolution de cet état émotionnel ni sur sa façon d'impacter le comportement.

L'autre approche est celle de la théorie de l'évaluation cognitive des émotions qui propose de lier cognition et émotion. La théorie développée par Smith et Lazarus [SL+90] a été utilisée par l'architecture EMA [GM04] mais elle reste compliquée à intégrer à une architecture comportementale globale à cause de ses variables d'évaluation subjectives. Le modèle OCC [OCC90] a par contre été développé pour être utilisable en IA et il a déjà été formalisé avec les termes du modèle BDI [Ada07].

Puisque la base cognitive de l'architecture développée est assurée par une architecture BDI, il parait naturel d'y intégrer les émotions via le modèle OCC et la formalisation qui en a été proposée à travers le modèle BDI [Ada07]. De cette façon, les émotions peuvent être considérer en parallèle des états mentaux cognitifs afin de modifier la prise de décision d'un agent.

Personnalité

Il existe deux modèles de représentation de la personnalité. D'une part, le modèle MBTI [MMM85] définit quatre critères possédant chacun deux valeurs possibles, menant à la définition de seize profils de personnalités distincts. En offrant un nombre fini de personnalités bien définies, ce modèle permet de simplement caractériser la personnalité d'une personne en la plaçant dans l'un des seize profils. Par contre, ce modèle ne permet pas de définir de très petites différences de personnalités.

Le modèle OCEAN [MJ92] a les caractéristiques opposées du modèle MBTI. Cinq dimensions continues permettent de représenter la personnalité comme un point dans un espace continue à cinq dimensions. Ainsi, de très petites différences entre deux personnalités peuvent être exprimées mais il devient plus difficile de caractériser des catégories de personnalités.

Dans les travaux ayant utilisé la personnalité pour des simulations sociales [OSC09b] [LLB12], c'est toujours le modèle OCEAN qui a été sélectionné. Il permet notamment de caractériser la personnalité par des valeurs numériques, le rendant plus simple à intégrer dans un système informatique. Pour cette raison, l'architecture développée dans cette thèse représente la personnalité d'un agent selon le modèle OCEAN.

Contagion émotionnelle

La contagion émotionnelle ne fait pas de débats au sein de la communauté des sciences cognitives. Il s'agit d'un processus modifiant les émotions d'une personne en fonction des émotions perçues parmi les autres personnes proches. Ce processus peut agir sur une émotion équivalente à celle qui est perçue (effet de fou rire où la joie influence la joie) ou sur une autre émotion que celle qui est perçue (effet de la personne triste par humiliation).

Le modèle ASCRIBE [BDM+09] se propose de modéliser ce processus en tenant compte des émotions en jeu, de la capacité à contaminer émotionnellement et à être contaminé émotionnellement, ainsi que d'une valeur représentant le canal de communication entre les personnes impliquées. Ce modèle est adapté dans l'architecture développée dans cette thèse, afin de correspondre à la représentation des émotions développée ainsi que pour être calculable sur des centaines d'agents.

Relations sociales

La section 2.3.2 montre qu'il existe deux façons de traiter les relations sociales. Une première façon consiste à donner un rôle institutionnel aux agents [RS62]. L'inconvénient de cette méthode est qu'il faut définir a priori l'ensemble des rôles possibles ainsi que leurs implications sur la prise de décision. De plus, cette représentation ne permet pas de définir finement des relations qui seraient créées dynamiquement hors de tout contexte institutionnel.

L'autre façon de traiter les relations sociales consiste à se baser sur le modèle dimensionnel des relations interpersonnelles [Sve00] qui indique qu'une relation sociale avec une autre personne se caractérise avec un nombre fini de variables à valeur bornée. Ce modèle de représentation des relations sociales est adapté à une architecture générique car elle n'impose pas de contexte social à prendre en compte, ce qui n'a pas toujours de sens en simulation sociale. De plus, la représentation sous la forme de valeurs numériques est adaptée à une implémentation dans un système informatique avec une évolution fine possible au cours du temps.

Si le modèle dimensionnel des relations interpersonnelles donne une représentation des relations sociale, il ne dit rien concernant leur évolution dans le temps ou leur apparition dynamique. Ces deux processus doivent être inclus dans l'architecture développée pour rendre le comportement agent dynamique et capable de s'adapter aux modifications en cours de simulation, y compris les modifications sociales.

Normes

En SMA, la création de système normatif est surtout passée par une implémentation de la logique déontique, comme le montre la section 1.3.2. L'inconvénient d'une telle méthode est qu'elle implique de transformer l'ensemble du raisonnement de l'agent selon cette logique. Cela rentrerait alors en conflit avec la partie cognitive de l'architecture, fondée sur le modèle BDI et la logique modale de Cohen et Levesque [CL90].

D'autres modèles ont été proposés comme BOID [BDH+01] qui ajoute un concept d'obligation agissant en parallèle des désirs de l'agent. Toutefois, cette architecture ne prend pas en compte les nombreuses distinctions entre les différents types de normes, comme expliqué en section 2.3.3. Lopez y Lopez [yLLd06] contourne le problème en rassemblant toutes les définitions de normes sous une seule formalisation. Pour autant, une telle approche impose de nouveau de considérer l'ensemble du comportement de l'agent comme une suite de normes.

Pour prendre en compte les différentes définitions et compréhension du concept de norme, et notamment permettre de faire une différence entre des obligations imposées et sanctionnées explicitement par une institution et des normes sociales imposées et sanctionnées implicitement par un groupe d'agents, le modèle normatif développé dans l'architecture de cette thèse s'inspire principalement de BOID. Ainsi, les concepts centraux à une architecture BDI sont doublés de concepts normatifs. C'est ensuite le moteur de raisonnement qui se charge de choisir entre les normes à suivre et les désirs possibles de l'agent.

Deuxième partie

Contributions

Chapitre 3

Formaliser la prise de décision d'un agent social dans une simulation

Sommaire

La prise de décision chez une personne humaine est un processus complexe qui implique la prise en compte de différentes dimensions affectives et sociales en plus d'un processus cognitif de raisonnement, comme le montre la chapitre 2. Ce sont donc des notions importantes à prendre en compte dans l'optique de la simulation d'acteurs humains. Pour autant, le chapitre 1 montre qu'il n'existe pas d'architecture comportementale en SMA et en simulation sociales regroupant toutes ces notions tout en étant intégrée à une plateforme de simulation.

L'architecture comportementale développée dans cette thèse, nommée BEN (Behavior with Emotions and Norms), intègre la cognition, la personnalité, les émotions, la contagion émotionnelle, la gestion des normes et les relations sociales pour le développement d'agents simulant des acteurs humains. Une formalisation de chacune de ces notions, définies dans le chapitre 2, doit être proposée afin de les inclure dans une même architecture. Cette formalisation permet de définir clairement la représentation de ces concepts afin de définir par la suite des règles d'interactions, dans l'optique de créer un comportement complexe et dynamique.

Ce chapitre expose la représentation des éléments manipulés par BEN. Cette architecture a pour ambition d'être intégrée dans des plateformes de simulation à base d'agent pour faciliter le travail de modélisateurs qui souhaiteraient simuler des acteurs humains. Il est donc important que la représentation des concepts manipulés soit simple à comprendre et à s'approprier. Pour cela, des formalismes déjà existants ont été retravaillés pour être homogénéisés entre eux , rendant l'appropriation de l'architecture plus rapide.

Une première section décrit les concepts manipulés par le raisonnement cognitif de l'architecture. Une deuxième section revient sur les dimensions affectives de l'architecture, à savoir la personnalité, les émotions et la contagion émotionnelle, traitée comme un processus lié aux émotions et faisant le lien avec la troisième section, centrée sur les dimensions sociales de l'architecture que sont la gestion des normes et les relations sociales.

3.1 Raisonner sur le monde

La première étape pour rendre le comportement des agents plus proche du comportement humain consiste à raisonner sur l'environnement. Cela signifie non seulement posséder une représentation au moins partielle du monde, mais aussi posséder la possibilité de prendre une décision sur un grand ensemble d'informations gardées en mémoires. L'intérêt est de ne pas simplement répondre à un changement brutal et ponctuel de l'environnement, mais plutôt de conserver une inertie dans le comportement, amenant l'agent à réaliser des actions complexes en s'adaptant progressivement au monde.

Pour effectuer ce raisonnement cognitif, l'agent a besoin, d'une part, de posséder une représentation unifier des éléments de l'environnement, et d'autre part, de posséder des concepts de haut niveaux permettant d'exprimer des états mentaux sur ces informations, permettant alors de prendre des décisions sur le long terme.

3.1.1 Représenter le monde avec des prédicats

La cognition est la partie centrale de BEN sur laquelle vont se greffer les autres dimensions. Un agent cognitif raisonne à partir d'informations venant, d'une part de ses perceptions de l'environnement et d'autre part de ses connaissances précédemment stockées en mémoire. Ces informations peuvent être des faits généraux, des situations ou des actions réalisées par d'autres agents et servent à décrire l'environnement. Pour représenter ces informations, BEN repose sur le concept de **prédicats**.

Un prédicat représente une information du monde, que ce soit un événement, une situation ou une action et s'écrit $\mathbf{P}_j\mathbf{(v)}$ dans sa forme générale avec les éléments suivants :

- **P** : l'identifiant du prédicat.
- **j** : l'agent causant l'information.
- **v** : l'ensemble de valeurs stockées dans le prédicat, sous la forme d'une liste associant une chaîne de caractère à la valeur stockée écrite de la façon suivante : $["valeur1" :: v1; "valeur2" :: v2; ...]$.

Cette forme générale peut s'adapter au contexte. Ainsi, un prédicat ne contenant aucune information particulière sera noté $\mathbf{P}_j$, un prédicat n'étant pas causé par un agent en particulier sera noté **P(v)** et un prédicat n'ayant aucune valeur particulière stockée et n'étant pas causé par un agent en particulier sera noté **P**. Enfin, la négation d'un prédicat P est notée **non P** et signifie que si P est vrai, non P est faux. Il s'agit donc d'une négation forte qui peut porter sur tous les prédicats possibles, qu'ils soient causés par un agent ou non, et qu'ils soient sur un ensemble de valeur particulière ou non.

> Pour prendre un exemple, l'information qu'il y a un incendie est notée feu, l'information selon laquelle l'incendie est précisément dans le bâtiment A est notée $feu(["emplacement" :: batimentA])$ et l'information selon laquelle l'agent b a mis le feu au bâtiment A est noté $feu_b(["emplacement" :: batimentA])$. En fonction de son écriture, un prédicat représente donc un événement général (comme le prédicat feu), une situation avec des valeurs précises (comme le prédicat $feu(["emplacement" :: batimentA])$) ou une action réalisée par un agent donné (comme le prédicat $feu_b(["emplacement" :: batimentA])$).

3.1.2 États mentaux et cognition

États mentaux cognitifs

Pour effectuer un raisonnement, et prendre une décision, l'agent suivant l'architecture BEN manipule des état mentaux cognitifs. Un état mental cognitif possédé par un agent i est représenté par $\mathbf{M}_i\mathbf{(P,V,L)}$ avec les éléments suivants :

— **M** : la modalité indiquant le type de l'état mental cognitif (par exemple une croyance).

— **P** : l'objet sur lequel porte l'état mental cognitif, pouvant être un prédicat, un autre état mental cognitif ou une émotion.

— **V** : une valeur réelle dont la signification dépend de la modalité. Permet de comparer deux états mentaux cognitifs de même modalité.

— **L** : une durée de vie indiquant le temps que mettra l'agent pour oublier cet état mental.

Comme pour les prédicats, les états mentaux cognitifs peuvent être écrits plus simplement en fonction du contexte. Un état mental cognitif ne possédant pas de durée de vie particulière et de valeur particulière s'écrit $\mathbf{M}_i\mathbf{(P)}$. Sans durée de vie particulière, un état mental cognitif n'est jamais oublié par l'agent. L'écriture $\mathbf{V[M}_i\mathbf{(P,V,L)]}$ représente la valeur attachée à l'état mental M_i(P,V,L) et l'écriture $\mathbf{L[M}_i\mathbf{(P,V,L)]}$ représente la durée de vie attachée à ce même état mental.

Dans BEN, la cognition se base sur le modèle BDI [Bra87] qui spécifie qu'un agent possède des croyances, des désirs et des intentions, représentant ses états mentaux stockés. Pour permettre de lier cette partie cognitive de l'architecture aux dimensions affectives et sociales, l'architecture définit 6 modalités différentes :

— $\mathbf{Belief}_i\mathbf{(P)}$: l'agent a une croyance sur P. Cela représente ce que l'agent croit sur le monde. La valeur attachée à cette croyance représente la force qui lui est accordée par l'agent.

— $\mathbf{Uncertainty}_i\mathbf{(P)}$: l'agent a une incertitude sur P. Cela représente une information incertaine sur le monde. La valeur attachée à cette incertitude représente son importance du point de vue de l'agent.

— $\mathbf{Desire}_i\mathbf{(P)}$: l'agent a un désir à propos de P. Cela représente un état du monde que l'agent souhaite atteindre. La valeur attachée à ce désir représente sa priorité par rapport aux autres désirs.

— $\mathbf{Intention}_i\mathbf{(P)}$: l'agent a une intention sur P. Cela représente un état du monde que l'agent s'engage à atteindre. La valeur attachée à cette intention représente sa priorité par rapport aux autres intentions.

— $\mathbf{Ideal}_i\mathbf{(P)}$: l'agent a un idéal a propos de P. Cela représente une information sur laquelle l'agent porte un jugement social. La valeur attachée à cet idéal représente la valeur morale accordée à P. Celle-ci peut être positive (P est louable) ou négative (P est blâmable).

— $\mathbf{Obligation}_i\mathbf{(P)}$: l'agent a une obligation sur P. Cela représente un état du monde que l'agent doit atteindre. La valeur attachée à cette obligation représente sa priorité par rapport aux autres obligations.

Les croyances de l'agent représentent des informations de l'environnement du point de vue de l'agent. Cela signifie que ce ne sont pas nécessairement des informations vraies et absolues. Pour autant, l'agent est sûr de ses croyances, contrairement aux incertitudes, qui représentent des informations dont l'agent a connaissance mais sur lesquels il ne possède pas de certitudes. Par exemple, si un agent observe de la fumée, il aura l'incertitude qu'il y a un incendie tout en ayant la croyance qu'il voit de la fumée.

Ainsi, la valeur attachée aux croyances ne représente pas un degré de vérité. Cette valeur représente un force accordée à la croyance par l'agent. Cela signifie qu'une croyance avec peu d'importance pour l'agent aura une faible valeur de force alors qu'une croyance jugée comme importante aura une haute valeur de force. De même, la valeur attachée à l'incertitude ne représente pas un degré de certitude sur l'information qui, s'il passait à 100%, amènerait cette incertitude à devenir une croyance. C'est là aussi une force indiquant l'importance donnée par l'agent à l'incertitude.

Les désirs, obligations et intentions sont en lien avec la prise de décision active de l'agent. Les désirs représentent les états du monde que l'agent souhaite lui-même voir atteint quand les obligations sont imposées par une institution. Le concept d'intention permet d'engager l'agent dans la tâche d'atteindre un état du monde, que cette tâche vienne précédemment des désirs ou des obligations, lui permettant de ne pas délibérer continuellement entre ses désirs et ses obligations.

Pour reprendre notre exemple sur le feu, $Belief_i(feu, 0.8, 15)$ représente la croyance de l'agent i qu'il y a un feu. L'agent i accorde à cette croyance une force de 0.8 et oubliera cette information dans 15 étapes de calcul. $Desire_i(feu)$ représente le désir de l'agent i qu'il y ait un feu, et ce désir ne sera pas oublié car il ne possède pas de durée de vie.

Plan d'action

Afin d'agir sur le monde en accord avec ses intentions, l'agent a besoin d'un plan d'actions sous la forme d'un ensemble de comportements exécutés dans un contexte particulier en réponse à une intention. Dans BEN, un plan possédé par l'agent i s'écrit $\mathbf{P}_i$**(I,C,Pr,B)** avec les éléments suivants :

— **P** : le nom du plan.

— **I** : le prédicat de l'intention déclenchant le plan.

— **C** : le contexte dans lequel le plan est applicable.

— **Pr** : une valeur de priorité utilisée pour choisir entre plusieurs plans applicables en même temps.

— **B** : le comportement, sous la forme d'une séquence d'instructions, à exécuter si le plan est sélectionné par l'agent.

Dans BEN, l'agent possède une base de plans, avec au moins un plan par intention différente possible (c.a.d des intentions portant sur différents objets). Si plusieurs plans répondent à la même intention, l'agent les départage en ne conservant dans sa délibération uniquement les plans dont le contexte est valide (c.a.d l'agent croit que le contexte du plan est vrai). Si plusieurs plans répondent à la même intention et possèdent des contextes valides, ils sont alors départagés par une valeur de priorité.

3.2 Éléments de psychologie affective

Avec l'architecture BEN, un agent dispose d'une personnalité et d'émotions en tant que dimensions affectives pour prendre une décision. Le chapitre 2 montre que l'ajout d'émotions dans le comportement d'agents informatique s'est appuyé sur le modèle OCC [OCC90] qui a été formalisé à travers le modèle BDI [Ada07] alors que l'utilisation de la personnalité s'est faite à travers le modèle OCEAN [MJ92] qui représente un consensus dans la représentation de la personnalité [Eys91]. La définition de ces notions s'appuie sur les formalismes proposés dans l'état de l'art et discutés dans la section 2.2.

3.2.1 Personnalité

La définition de traits de personnalité dans BEN suit le modèle OCEAN [MJ92], aussi connu sous le nom FFM (Five Factor Model). La personnalité d'un agent est représentée par un vecteur de 5 valeurs réelles comprises entre 0 et 1, avec la valeur neutre placée à 0.5. Les 5 traits de personnalité sont les suivants :

— **O** : représente l'ouverture d'esprit de quelqu'un. Quelqu'un de fermé d'esprit a une valeur de 0, quelqu'un d'ouvert d'esprit a une valeur de 1.

— **C** : représente la capacité de préparation à la prise d'action d'une personne. Quelqu'un d'impulsif a une valeur de 0, quelqu'un qui agit avec préparation a une valeur de 1.

— **E** : représente l'extraversion d'une personne. Quelqu'un de timide a une valeur de 0, quelqu'un d'extraverti a une valeur de 1.

— **A** : représente la capacité de quelqu'un à être agréable. Quelqu'un ayant une attitude hostile envers les autres a une valeur de 0, quelqu'un d'amical a une valeur de 1.

— **N** : représente la capacité d'une personne à contrôler ses réactions émotionnelles. Quelqu'un de neurotique, qui n'arrive pas à contrôler ses émotions, a une valeur de 0, quelqu'un restant calme devant ses émotions a une valeur de 1.

Ainsi, un agent a une personnalité sous la forme $(0.2, 0.3, 0.4, 0.5, 0.6)$. Une personnalité neutre est représentée par le vecteur $(0.5, 0.5, 0.5, 0.5, 0.5)$, signifiant que l'agent est exactement dans la moyenne sur chaque trait de personnalité. Ces valeurs de personnalité sont initialisées au démarrage de la simulation et n'évoluent pas au cours de celle-ci.

3.2.2 Émotions

Représenter les émotions

Dans BEN, les émotions sont définies selon le modèle OCC [OCC90], détaillé en section 2.2.1, qui décrit une émotion comme une réponse valuée à l'évaluation cognitive d'une situation. Aussi, puisque les agents doivent prendre des décisions en fonction de leur contexte social, chaque émotion doit renseigner l'agent qui en est à l'origine, pour pouvoir agir en conséquence avec cet agent. Ainsi, une émotion est représentée par **E_i(P,A,I,D)** avec les éléments suivants :

— **E** : le nom de l'émotion.

— **P** : le prédicat à propos duquel l'émotion est ressentie.

— **A** : l'agent responsable de l'émotion.

— **I** : l'intensité de l'émotion. Cette valeur est positive ou nulle.

— **D** : la valeur de décroissance de l'intensité émotionnelle.

Les émotions peuvent avoir des écritures plus simples en fonction du contexte. Une émotion sans intensité particulière et sans valeur de décroissance particulière est représentée par **E_i(P,A)**, une émotion sans agent responsable particulier est représentée par **E_i(P,I,D)** et une émotion sans intensité particulière, sans valeur de décroissance particulière et sans agent responsable est représenté par **E_i(P)**. L'écriture **I[E_i(P,A,I,D)]** représente l'intensité de l'émotion E_i(P,A,I,D) et l'écriture **D[E_i(P,A,I,D)]** représente sa valeur de décroissance.

Par exemple, l'agent A qui ressent de la joie à cause du beau temps météorologique aura une émotion $Joie_{\mathrm{A}}(beauTemps)$. Si cette même émotion a une intensité de valeur 14 et de décroissance 2, cela s'écrira $Joie_{\mathrm{A}}(beauTemps, 14, 2)$. Enfin, si l'information du beau temps météorologique lui est donnée par l'agent B, sans que l'agent A n'ait réellement connaissance du temps météorologique, l'émotion s'écrira $Joie_{\mathrm{A}}(beauTemps, \mathrm{B}, 14, 2)$.

Dans l'architecture BEN, c'est la présence, ou non, d'une émotion dans la base des émotions de l'agent qui prévaut. Cela explique la possibilité d'avoir des émotions sans intensité, et donc sans valeur de décroissance. Cela implique aussi que la diminution de l'intensité émotionnelle se fait par soustraction de la valeur de décroissance à l'intensité. Si l'intensité devient négative, l'émotion est retirée de la mémoire de l'agent.

L'agent responsable de l'émotion peut soit être l'agent ayant causer l'information P sur laquelle porte l'émotion, soit être l'agent communicant l'information P. Par exemple, si l'agent A voit l'agent B mettre le feu à un bâtiment, l'émotion créée sera $peur_{\mathrm{A}}(feu_{\mathrm{B}}, \mathrm{B})$. Par contre, si l'agent A apprend par l'agent C que l'agent B a mis le feu au bâtiment, l'émotion de peur s'écrira $peur_{\mathrm{A}}(feu_{\mathrm{B}}, \mathrm{C})$.

Cette définition permet à un agent de ressentir en même temps plusieurs émotions qui sont classées en fonction de leur intensité. Cela signifie qu'un agent peut ressentir ponctuellement une grande peur, qui impactera immédiatement son comportement, mais que cette peur peut vite s'estomper pour laisser la place à une émotion de joie que l'agent ressentait précédemment et dont l'intensité, plus faible, a moins diminué dans le temps.

Définir la contagion émotionnelle

La contagion émotionnelle est un processus où les émotions d'un agent sont influencées par la perception des émotions des agents aux alentours [HCR93]. Cette influence peut se traduire par la copie de l'émotion perçue ou par la création d'une nouvelle émotion, comme indiqué dans la section 2.3.1.

Dans BEN, la contagion émotionnelle est formalisée à partir d'une version simplifiée du modèle ASCRIBE [BDM+09]. Une contagion émotionnelle entre un agent i contaminant et un agent j contaminé est décrite par $(\mathbf{E}_i,\mathbf{E}_j,\mathbf{C}_i,\mathbf{R}_j,\mathbf{T}_j)$ avec :

— $\mathbf{E}_i$: l'émotion de l'agent i qui déclenche la contagion si elle est perçue par l'agent j.

— $\mathbf{E}_j$: l'émotion créée par l'agent j à la suite de la contagion émotionnelle. Cela peut être une copie de E_i (avec d'autres valeurs d'intensité et de décroissance) ou une nouvelle émotion.

— $\mathbf{C}_i$: la valeur de charisme de l'agent$i$ indiquant sa capacité à transmettre ses émotions.

— $\mathbf{R}_j$: la valeur de réceptivité de l'agent j indiquant sa capacité à être influencé par les émotions des autres agents.

— $\mathbf{T}$: une valeur seuil. La contagion est exécutée uniquement si la multiplication $\mathrm{C}_i \times \mathrm{R}_j$ est supérieure à ce seuil.

Le modèle ASCRIBE inclut une valeur représentant le canal de communication entre l'agent contaminant et l'agent contaminé. Cette valeur est ramenée, par les auteurs, à un calcul de distance entre agents, jouant sur le fait que plus deux personnes seront éloignées, moins la contagion fera effet. Le problème se pose dans la généralisation de cette valeur par rapport au concept de calcul de distance. La valeur de seuil a pour but de remplacer cette valeur de canal entre les agents. C'est à la discrétion de l'utilisateur de mettre un seuil qui pourra, par exemple, être calculé en fonction d'une distance, définie lors de l'implémentation, entre les agents.

De plus, ce seuil permet d'indiquer les cas où la contagion ne se fait pas. Dans le modèle ASCRIBE, l'agent possède en permanence l'ensemble des émotions possibles, chacune exprimée par une valeur en 0 et 1. Cela signifie que même la plus petite contagion émotionnelle, c'est à dire une faible évolution de l'intensité chez l'agent contaminé, est prise en compte. Dans BEN, c'est la présence, ou non, d'émotions qui prévaut. Cela signifie que si la contagion est trop faible, sous le seuil fixé, elle n'a pas lieu, ne créant pas d'émotion chez l'agent contaminé.

3.3 Éléments de psychologie sociale

En suivant l'architecture BEN, un agent peut agir en fonction de son contexte social via une gestion des normes et la définition de relations sociales avec les autres agents. Ces notions, formalisées dans cette section, sont décrites en détail dans la section 2.3.

3.3.1 Normes, obligations et contrôles

Comme le montre la section 2.3.3, le concept de norme est vaste et complexe. Afin de permettre à chaque utilisateur de s'approprier la gestion des normes dans l'architecture BEN, plusieurs concepts ont été mis en place pour couvrir les différentes définitions des normes sociales et des obligations. Cette section introduit les concepts de "Norme", de "Loi" et de "Contrôle" qui permettent à un agent suivant l'architecture BEN, de prendre une décision en fonction d'un système normatif.

Normes

Dans BEN, la notion de norme s'appuie sur les travaux de Tuomela [Tuo95], sur l'architecture BOID [BDH+01] et sur le framework proposé par Lopez y Lopez [yLLd06]. Une norme, dans l'architecture BEN, est définie comme un comportement, actif dans un contexte précis, auquel

l'agent peut obéir pour répondre à l'une de ses intentions. Une norme possédée par l'agent i est représentée par $\mathbf{N}_i$**(I,C,O,Pr,B,V)** avec les éléments suivants :

- **N** : le nom de la norme.
- **I** : le prédicat de l'intention déclenchant la norme
- **C** : le contexte dans lequel la norme peut être appliquée.
- **O** : une valeur d'obéissance qui sert de seuil, déterminant si la norme est appliquée ou non, en fonction de la valeur d'obéissance de l'agent.
- **Pr** : une valeur de priorité utilisée pour choisir entre plusieurs normes applicables en même temps.
- **B** : le comportement, sous la forme d'une séquence d'instruction, à exécuter si la norme est suivie par l'agent.
- **V** : Une durée de violation indiquant le temps pendant lequel la norme est considérée violée une fois qu'elle a été violée.

La notion de norme définie ici est à mettre en parallèle des plans d'actions présentés en section 3.1.2. Il s'agit donc de comportements à appliquer en réponse à une intention de l'agent. À la différence des plans, qui sont des constructions personnelles de l'agent, les normes sont des constructions partagées par une population d'agents et implicitement imposées par cette population. C'est la raison pour laquelle les agents peuvent désobéir à ces normes, c'est à dire choisir ne pas les exécuter alors qu'elle réponde à une intention courante et que leur contexte est valide.

Cette formalisation d'une norme est suffisante pour couvrir le concept de normes sociales qui correspondent à un comportement imposé implicitement par un groupe d'agents s'appliquant dans un contexte précis et pouvant être violé. Pour autant, ce formalisme est aussi adapté à des conceptions plus larges que les stricts normes sociales puisque les normes, comme définies dans BEN, sont avant tout des comportements répondant à des intentions et pouvant ne pas être suivi par l'agent, donnant par la suite lieu à une sanction.

Un exemple de norme sociale serait le fait de marcher dans le même sens que la majorité des gens dans une foule. Une telle norme serait représentée, avec le formalisme, de la façon suivante :

- N : la chaîne de caractères "marcherDansLeMemeSens" comme nom de la norme.
- I : le prédicat $marcher$, représentant le fait que la norme n'est utiliser que lorsque l'agent à l'intention de marcher.
- C : le fait de se trouver dans une foule. Si cette information n'est pas vérifiée, la norme ne s'applique pas.
- O : 0.5 représentant le fait que l'agent doit avoir une valeur d'obéissance supérieur à 0.5 pour suivre la norme.
- Pr : 10.0 indique la priorité de la norme par rapport à d'autres normes activables dans les mêmes conditions.
- B : une séquence d'instruction indiquant à l'agent comment marcher dans le même sens que la majorité des agents de la foule, en s'appuyant sur des actions atomiques.
- V : 1, indiquant que la norme est considérée comme violée pendant 1 étape de calcul après sa violation.

Lois et obligations

La notion de "norme", comme elle est formalisée dans BEN, est suffisante pour couvrir le concept de norme sociale mais elle ne permet pas de pleinement représenter le concept d'obligation, où une institution impose un comportement à une personne. Pour cela, il faut définir la notion de "loi".

Une loi est une règle explicite, imposée à l'agent par une autorité, qui crée une obligation, en tant qu'état mental cognitif, dans un contexte précis. Cette loi peut toutefois être violée. Avec ces éléments, l'architecture BEN définit une loi par **L(C,Obl,O)** avec :

— **L** : le nom de la loi.

— **C** : le contexte dans lequel la loi doit être appliquée.

— **Obl** : le prédicat qui sera intégré dans l'obligation créée par la loi.

— **O** : une valeur d'obéissance servant de seuil pour savoir si la loi doit être exécutée, en fonction de la valeur d'obéissance de l'agent.

La définition d'une loi permet d'ajouter des obligations à l'agent en parallèle de ses désirs. Le processus de délibération permettra de choisir entre les obligations et les désirs pour créer les intentions de l'agent. Pour autant, pour répondre à une intention issue d'une obligation, l'agent devra posséder une norme indiquant le comportement à suivre suite à cette intention issue d'une obligation.

Par exemple, une loi peut représenter le fait qu'en France, l'État impose aux conducteurs de voiture de rouler sur la droite de la route. Cette loi serait formalisée de la façon suivante :

— L : la chaîne de caractère "roulerADroite" comme nom de la loi.

— C : le fait d'être automobiliste sur une route française.

— Obl : le prédicat $a\mathrm{D}roite$ sur lequel une obligation sera créée et ajoutée à l'agent dans le cas où la loi est exécutée.

— O : 0.1, indiquant que tous les agents possédant une valeur d'obéissance supérieur à 0.1 exécuterons la loi si le contexte est valide.

En plus de cette loi, il faudra définir une norme, permettant de répondre à l'intention issue de l'obligation créée par la loi. Cette norme aurait le formalisme suivant :

— N : la chaîne de caractères "conduireADroite" comme nom de la norme.

— I : le prédicat $a\mathrm{D}roite$.

— C : le fait d'être automobiliste sur une route française.

— O : 0.1 représentant le fait que l'agent doit avoir une valeur d'obéissance supérieur à 0.1 pour suivre la norme.

— Pr : 10.0 indique la priorité de la norme par rapport à d'autres normes activables dans les mêmes conditions.

— B : une séquence d'instruction indiquant à l'agent comment conduire son véhicule sur la droite de la route, en s'appuyant sur des actions atomiques.

— V : 1.

Contrôles

Puisque les normes et les lois peuvent être violées, l'architecture a besoin de définir un système de contrôle pour appliquer des sanctions et des récompenses, représentant des comportements de réaction face à la violation ou à l'application d'une norme ou d'une loi. Une sanction est une séquence d'instructions déclenchée lors du contrôle. Un contrôle effectué par un agent i sur un agent j est représenté par ($\mathbf{M}_j$,$\mathbf{S}_i$,$\mathbf{R}_i$) avec les éléments suivants :

— $\mathbf{M}_j$: la modalité de l'agent j qui doit être contrôlée. Cette modalité est soit une norme, soit une loi, soit une obligation.

— $\mathbf{S}_i$: la sanction que l'agent i applique si la modalité contrôlée est violée.

— $\mathbf{R}_i$: la sanction, appelée récompense, que l'agent i applique si la modalité contrôlée a été suivie.

Le contrôle est effectué différemment en fonction de la modalité contrôlée. Un contrôle par rapport à une norme consiste à regarder si la norme en question est applicable chez l'agent contrôlé. Si la norme est applicable et que l'agent contrôlé l'a appliquée, ce dernier peut recevoir une récompense. Par contre, si la norme était applicable mais que l'agent ne l'a pas appliquée, il peut être soumis à une sanction. Enfin, si la norme n'était pas applicable, le contrôle s'arrête.

Dans le cas du contrôle d'une loi, le fonctionnement est similaire au contrôle d'une norme. Si la loi était applicable pour l'agent contrôlé et qu'elle a été appliquée, alors la récompense peut être exécutée. À l'inverse, si la loi était applicable et que l'agent ne l'a pas appliquée, la sanction peut être déclenchée. Enfin, si la loi n'est pas applicable, elle n'est pas soumise au contrôle.

Enfin, le contrôle d'une obligation permet de contrôler le fait qu'un agent a bien choisi une norme répondant a ses obligations. Cela signifie qu'un agent peut suivre une loi, créer l'obligation mais ne pas réaliser le comportement imposé par cette obligation. Cela signifie que l'agent a conscience de la loi mais préfère suivre ses désirs personnels. Le contrôle se fait en vérifiant la base des obligations de l'agent contrôlé et en regardant si la norme utilisée répond à cette obligation. Si c'est le cas, la récompense est appliquée, sinon c'est la sanction qui est appliquée. La sanction, ou la récompense, d'un agent en fonction de ses obligations n'est faite qu'une seule fois jusqu'à ce qu'il reconsidère ses plans pour ne pas pénaliser continuellement un agent qui n'a pas réévalué ses décisions.

Les sanctions et récompenses sont toujours définies par l'agent contrôlant. Ce sont des séquences d'instructions à effectuer dans le cas d'une violation de norme, pour la sanction, ou dans le cas d'une norme correctement suivie, dans le cas d'une récompense. Ces deux informations ne sont pas obligatoires ; il est possible de définir un contrôle sans sanction ou sans récompense. Aussi, ces sanctions peuvent êtres des actions explicites envers l'agent contrôlé, comme une amende à payer, ou implicite, comme une modification de la relation sociale avec l'agent contrôlé.

La définition de contrôles permet de donner du sens aux normes et aux lois définies dans le modèle et ainsi de couvrir l'ensemble des définition de la notion de norme expliquée en section 2.3.3. De cette façon, chaque utilisateur définit un système normatif qui contraindra la prise de décision de l'agent,quelque soit sa définition de la notion de norme.

3.3.2 Relations sociales

L'architecture BEN permet de créer des relations sociales entre les agents afin de modifier leur comportement en fonction du contexte social. En s'appuyant sur les travaux de Svennevig [Sve00], les relations sociales sont décrites par un nombre fini de variables. Svennevig identifie un ensemble minimal de quatre variables, présentées en détail en section 2.3.2 : l'appréciation, la dominance, la solidarité et la familiarité. Dans BEN, une cinquième variable représentant la confiance, est ajoutée pour permettre une interaction avec le système normatif via l'évolution de la confiance en fonction des sanctions et des récompenses sociales. Ainsi, une relation sociale, dans BEN, possédée par un agent i envers un agent j est représentée par $\mathbf{R}_{i,j}$**(L,D,S,F,T)** avec les éléments suivants :

— **R** : l'identifiant de la relation sociale.

— **L** : une valeur réelle entre -1 et 1 représentant le degré d'appréciation avec l'agent concerné par le lien. Une valeur de -1 indique que l'agent j est détesté, une valeur de 1 indique que l'agent j est adoré.

— **D** : une valeur réelle entre -1 et 1 représentant le degré de pouvoir exercé sur l'agent concerné par le lien. Une valeur de -1 indique que l'agent j domine la relation, une valeur de 1 indique que l'agent j est dominé dans la relation.

— **S** : une valeur réelle entre 0 et 1 représentant le degré de solidarité avec l'agent concerné par le lien. Une valeur de 0 indique qu'il n'y a aucune solidarité envers l'agent j, une valeur de 1 indique une solidarité totale avec l'agent j.

— **F** : une valeur réelle entre 0 et 1 représentant le degré de familiarité avec l'agent concerné par le lien. Une valeur de 0 indique qu'il n'y a aucune familiarité avec l'agent j, une valeur de 1 indique qu'il y a une familiarité totale avec l'agent j.

— **T** : une valeur réelle entre -1 et 1 représentant le degré de confiance avec l'agent concerné. Une valeur de -1 indique que l'agent i n'a aucune confiance envers l'agent j, une valeur de 1 indique une totale confiance envers l'agent j.

Avec cette définition, il convient de préciser qu'une relation sociale n'est pas nécessairement symétrique : $R_{i,j}$(L,D,S,F,T) n'est pas égal par définition à $R_{j,i}$(L',D',S',F',T'). Cela s'explique par le fait que les valeurs de la relations sont évaluées par l'agent lui-même. Un agent A peut donc apprécier un agent B qui, en retour, déteste A.

L'écriture **L**[$\mathbf{R}_{i,j}$] représente la valeur d'appréciation de la relation sociale entre les agent i et j, l'écriture **D**[$\mathbf{R}_{i,j}$] représente sa valeur de dominance, l'écriture **S**[$\mathbf{R}_{i,j}$] représente représente sa valeur de solidarité, l'écriture **F**[$\mathbf{R}_{i,j}$] représente sa valeur de familiarité et l'écriture **T**[$\mathbf{R}_{i,j}$] représente sa valeur de confiance.

Par exemple, un agent i peut apprécier un agent j avec une valeur de 0.4, le dominer avec une valeur de 0.8, être solidaire avec une valeur de 0.2, être familier avec une valeur de 0.3 et avoir confiance avec une valeur de 0.5. Cette relation social sera représentée par $R_{i,j}(0.4, 0.8, 0.2, 0.3, 0.5)$. De l'autre coté, l'agent j peut apprécier l'agent i avec une valeur de -0.3 (ce qui signifie que l'agent j n'apprécie pas l'agent i), être dominé par i avec une valeur de 0.2, ne pas être solidaire du tout, être entièrement familier et avoir une valeur de confiance de -0.4. Cette relation sociale sera représentée par $R_{j,i}(-0.3, -0.2, 0.0, 1.0, -0.4)$.

3.4 Conclusion

L'architecture BEN développée dans cette thèse propose un cadre pour développer des agents sociaux ayant des capacités cognitives, une personnalité, des émotions, de la contagion émotionnelle, une gestion des normes et des relations sociales. Ce chapitre a présenté la formalisation permettant de représenter ces différentes notions.

Ce formalisme distingue, d'une part, les éléments manipulables par l'agent comme les états mentaux cognitifs, les plans d'action, les émotions, les normes, les lois et les relations sociales et d'autre part, les processus nécessitant une formalisation, comme la contagion émotionnelle ou le contrôle des normes. Enfin, la personnalité possède sa propre représentation puisque c'est la seule dimension sur laquelle l'agent n'a aucun contrôle.

L'uniformisation des représentations a pour but de rendre l'appropriation de l'architecture plus simple du point de vue utilisateur. Les éléments manipulés ont ainsi tous une forme globale similaire et où seul change le nombre de paramètres possibles pour chacun d'entre eux. De plus, les multiples écritures proposées pour un même concept mettent en avant l'aspect adaptatif de l'architecture. Ainsi, si un utilisateur estime ne pas avoir besoin de l'intensité émotionnelle, il est possible de ne pas l'utiliser dans la définition des émotions.

Ce formalisme est au cœur de l'architecture BEN détaillée dans le chapitre 4 et permet de modéliser un agent social comme un ensemble de composants de hauts niveaux. Chacun de ces comportements peut alors jouer un rôle dans le raisonnement et la prise de décision de l'agent, notamment en créant un contexte social qui complète le contexte environnemental perçu.

Chapitre 4

BEN : une architecture pour des agents au comportement cognitif, affectif et social

Sommaire

Ce chapitre présente l'architecture comportementale développée pour permettre la simulation informatique d'acteurs humains en tenant compte de facteurs cognitifs, affectifs et sociaux. Pour cela, l'architecture BEN (Behavior whith Emotions and Norms), intègre les concepts de cognition, d'émotion, de personnalité, de contagion émotionnelle, de gestion des normes et de relations sociales telles qu'ils sont discutés dans le chapitre 2. Ces concepts sont utilisés selon le formalisme présenté dans le chapitre 3 dans une architecture comportementale offrant un comportement dynamique, cognitif, affectif et social aux agents simulés.

Comme expliqué dans le chapitre 1, les simulations sociales visent à reproduire le comportement d'acteurs humains. Pour ce faire, une approche consiste à mimer le plus fidèlement possible les processus de prise de décision. Pour autant, toutes les dimensions de la prise de décision ne sont pas à forcément à considérer en même temps selon la situation étudiée. Dans certains cas, l'utilisation d'émotions, de relations sociales ou de normes peut ne pas avoir de sens ou être compliquée à intégrer dans un modèle par manque de données sur ces notions.

L'architecture BEN, représentée par la figure 4.2, offre un ensemble de processus permettant de créer un comportement pour l'agent s'adaptant à la dynamique de l'environnement. BEN se compose de 4 modules contenant les processus d'évolution des données, connectés aux bases de connaissances de l'agent. Chaque module propose des processus gérés automatiquement par l'architecture (en bleu sur la figure 4.2) ou devant être définis manuellement par le modélisateur (en rose sur la figure 4.2). Certains de ces processus sont obligatoires dans le fonctionnement de l'architecture (en lignes pleines sur la figure 4.2), les autres sont optionnels (en pointillés sur la figure 4.2). Enfin, l'ensemble est paramétré par la personnalité OCEAN de l'agent, qui n'est pour autant pas obligatoire pour le fonctionnement de l'architecture.

La figure 4.1 montre un diagramme d'activité de l'architecture BEN, donnant ainsi l'ordre d'exécution de chaque module lorsqu'un agent doit prendre une décision en utilisant cette archi-

tecture. Le code visuel utilisé est le même que pour la figure 4.2, permettant de mettre en évidence quelles sont les parties obligatoires ou non, et quels sont les processus à définir manuellement par l'utilisateur.

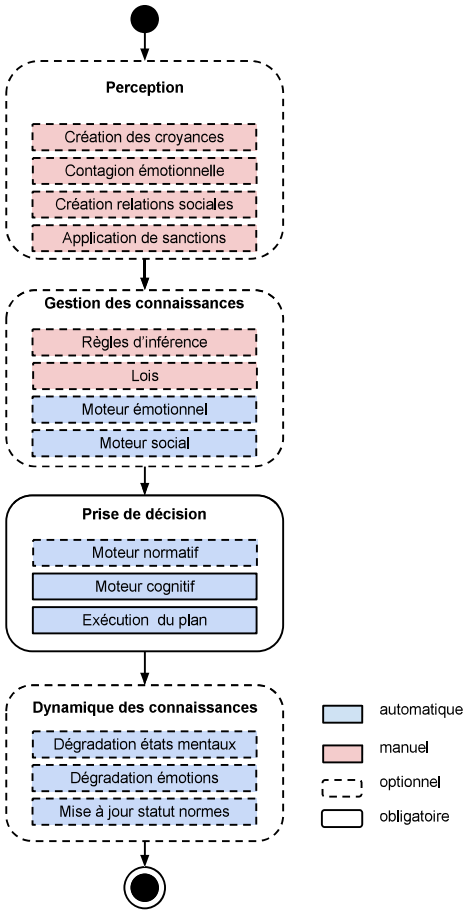

FIGURE 4.1 – Diagramme d'activité de l'architecture BEN

Dans la suite de ce chapitre, une première section sert d'introduction générale à l'architecture, avant d'expliquer le fonctionnement détaillé de chaque processus dans une deuxième section. Une dernière section discute des choix effectués pour réaliser cette architecture avant de conclure le chapitre.

4.1 Vue générale de l'architecture BEN

Le formalisme proposé dans le chapitre 3 permet de représenter les éléments manipulés par BEN de manière statique. Pourtant, ces éléments sont amenés à évoluer au cours de la simulation afin de créer un comportement s'adaptant aux changements de l'environnement. Le formalisme peut alors paraître lourd à retranscrire dans une architecture de comportement et compliqué à utiliser. À cela s'ajoute la difficulté de paramétrer un tel système sans pour autant avoir toujours les données nécessaires, en matière d'émotions ou de relations sociales par exemple.

Pour résoudre ces contraintes, l'architecture BEN est modulaire et propose de ramener les paramètres des différents processus à des calculs automatiques s'appuyant sur la personnalité. Ces deux notions sont détaillées dans le reste

4.1.1 Modularité de l'architecture

Dans BEN, tous les processus et composants apparaissant sont optionnels, sauf le moteur cognitif ainsi que les bases cognitives et la base des plans qui y sont rattachées, comme le montre la

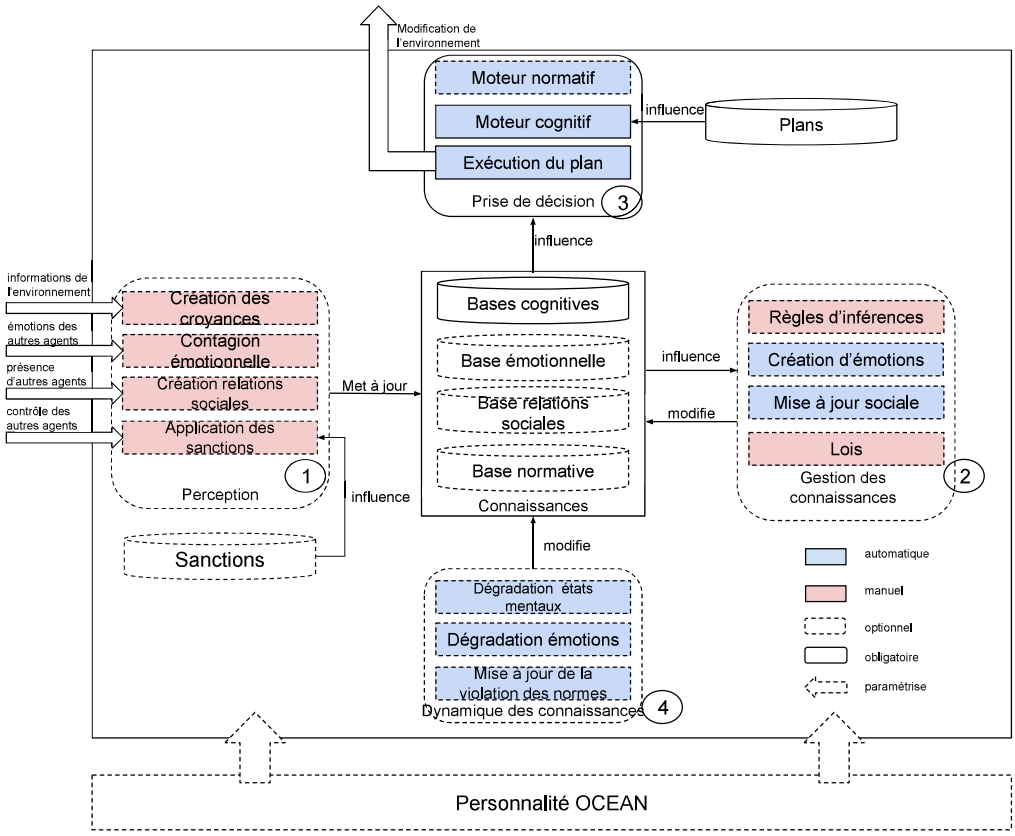

FIGURE 4.2 – Schéma de l'architecture BEN

figure 4.2. Cette modularité permet à chaque modélisateur de n'utiliser que les composantes nécessaires à la simulation de la situation étudiée, et donc d'éviter des calculs inutiles dans le cas considéré. Les processus et les dimensions psychologiques intégrées dans l'architecture sont interconnectés, c'est à dire qu'ils travaillent ensembles pour définir le comportement des agents, mais ils ne sont pas inter-dépendant. Le but est que l'utilisation soit la plus transparente possible du point de vue du modélisateur. L'architecture fonctionne donc quelques soient les processus optionnels activés ou désactivés.

Cette modularité permet de définir le comportement de l'agent modélisé de façon incrémentale. L'utilisateur peut commencer par spécifier les éléments obligatoires de l'architecture, c'est-à-dire les désirs et les plans, puis ajouter au fur et à mesure des processus, d'abord cognitifs puis affectifs ou sociaux. Cette modélisation incrémentale facilite l'utilisation de BEN puisque, si un processus est estimé trop complexe par un utilisateur ou non adapté à son cas d'étude, il n'est pas obligé d'être utilisé.

Pour rendre l'ensemble de ces processus indépendants les uns des autres sans altérer le fonctionnement global de l'architecture, il faut les relier à des bases de connaissances distinctes les unes des autres mais au centre de toute l'architecture. Ainsi, un processus utilisé va pouvoir se connecter et agir avec toutes les connaissances de l'agent, en n'utilisant seulement que les connaissances nécessaires et sans interférer avec les autres processus.

Les connaissances de l'agent se compose des bases suivantes :

— Les bases cognitives qui contiennent tous les états mentaux de l'agent, détaillés et formalisés en section 3.1.2.

— La base émotionnelle qui contient les émotions de l'agent, formalisées en section 3.2.2.

— La base des relations sociales qui contient l'ensemble des relations sociales avec les autres agents, formalisées en section 3.3.2.

— La base normative qui contient les normes de l'agent, formalisées en section 3.3.1.

En plus de ces connaissances centrales qui peuvent être modifiées au cours de la simulation par les processus de l'architecture, un agent dont le comportement est défini par BEN dispose de deux bases qui ne sont pas impactées par les différents processus :

— La base des plans qui contient les plans d'action permettant à l'agent d'agir sur l'environnement, suivant le formalisme détaillé en section 3.1.2.

— La base des sanctions qui contient l'ensemble des sanctions que l'agent peut utiliser lors du contrôle des normes, comme formalisé en section 3.3.1.

Finalement, l'ensemble de l'architecture repose sur la personnalité de l'agent, formalisé suivant le modèle OCEAN comme détaillé en section 3.2.1. Cette personnalité est fixe sur la durée de la simulation et permet de paramétrer les différents processus de BEN.

4.1.2 Paramétrer le comportement avec la personnalité

En plus des bases de connaissances, l'agent possède des variables liées aux dimensions cognitives, sociales et affectives de son comportement :

— La probabilité de conserver le plan en cours, qui permet à l'agent de remettre en cause son plan courant de façon aléatoire et ainsi de vérifier s'il n'existe pas un meilleur plan dans le contexte actuel.

— La probabilité de conserver l'intention en cours, qui permet à l'agent de reconsidérer son raisonnement actuel de façon aléatoire.

— Une valeur de charisme reliée au processus de contagion émotionnelle.

— Une valeur de réceptivité émotionnelle reliée au processus de contagion émotionnelle.

— Une valeur d'obéissance liée à la prise de décision normative.

Pour faciliter l'initialisation de ces variables et des paramètres liés à la création des émotions et à la mise à jour des relations sociales, détaillés respectivement en section 4.2.2 et en section 4.2.2, du point de vue du modélisateur, chaque valeur à calculer est reliée à au moins un trait de la personnalité, fondée sur le modèle OCEAN [MJ92] et formalisée en section 3.2.1. Ainsi, le nombre de paramètres que l'utilisateur a besoin d'entrer pour paramétrer le comportement des agents modélisés est réduit aux cinq dimensions du modèle OCEAN.

En lien avec la cognition, l'agent possède deux paramètres représentant la probabilité de remettre en cause aléatoirement le plan en cours ou l'intention courante, dans le but de vérifier si l'agent doit persévérer, ou non, dans son action en cours, alors que le contexte peut avoir été modifié. Ces deux valeurs sont liées à la composante de conscience du modèle OCEAN, qui décrit la tendance d'une personne à préparer ses actions (avec une valeur proche de 1) ou à agir de façon impulsive, en improvisant (avec une valeur proche de 0). L'équation (4.1) décrit l'initialisation de la probabilité de conserver le plan en cours et l'équation (4.2) décrit l'initialisation de la probabilité de conserver l'intention en cours.

$$Probabilité Conserver Plan = \sqrt{C} \tag{4.1}$$

$$Probabilité Conserver Intention = \sqrt{C} \tag{4.2}$$

Le formalisme du processus de contagion émotionnelle, décrit en section 3.2.2, implique la définition de valeurs de charisme et de réceptivité émotionnelle pour chaque agent. Dans BEN, le charisme, qui est la capacité d'une personne à exprimer ses émotions, est lié à la capacité d'expression du modèle OCEAN, qui décrit la capacité de quelqu'un à s'exprimer. La valeur de réceptivité émotionnelle est liée à la capacité de contrôle des émotions, représentée par la dimension neurotique du modèle OCEAN, où une valeur proche de 0 indique quelqu'un qui est très impacté par ses émotions alors qu'une valeur proche de 1 indique quelqu'un ayant le contrôle sur ses émotions.

L'équation (4.3) indique la formule de calcul du charisme et l'équation 4.4 indique la formule de calcul de la réceptivité émotionnelle.

$$charisme = \mathrm{E} \tag{4.3}$$

$$réceptivité = 1 - \mathrm{N} \tag{4.4}$$

La gestion des normes implique une valeur d'obéissance entre 0 et 1, indiquant la tendance de l'agent à suivre les lois, les obligations et les normes. Begue [BBC+15] a étudié la notion d'obéissance dans une recréation de l'expérience de Milgram [Mil63] : des sujets de test ont été placés dans un contexte d'émission télévisée où il fallait envoyer des décharges électriques à un second candidat, incarné par un acteur, lorsqu'il se trompait de réponse. Plusieurs semaines après ce test initial, les candidats ont répondu à un questionnaire déterminant leur personnalité selon le modèle OCEAN. L'étude statistique des différentes personnalités en parallèle des niveaux d'obéissance de chacun a conduit les auteurs à considérer que les dimensions de conscience et d'agréabilité du modèle OCEAN permettent de caractériser l'obéissance d'une personne. L'équation (4.5) indique le calcul effectué dans BEN pour initialiser la valeur d'obéissance d'un agent.

$$obéissance = \sqrt{\frac{\mathrm{C} + \mathrm{A}}{2}} \tag{4.5}$$

Tous les paramètres nécessaires à chaque processus de l'architecture, développés dans le reste de la section, suivent le même principe que l'initialisation : ils sont reliés aux dimensions du modèle OCEAN pour limiter le nombre de paramètres à entrer de la part de l'utilisateur.

4.2 Processus d'évolution des connaissances de l'agent

L'ensemble des processus permettant de créer un comportement cognitif, affectif et social est détaillé dans cette section. Ils sont présentés module par module, avec la perception de l'environnement dans un premier temps, la gestion des connaissances ensuite, la prise de décision et enfin l'évolution dynamique des connaissances.

4.2.1 Module de perception

La première étape de BEN, correspondant au module numéroté 1 sur la figure 4.2, consiste en la perception de l'environnement. Ce module est utilisé pour connecter l'environnement et les connaissances de l'agent, en transformant les informations venant du monde en états cognitifs mentaux, en émotions ou en liens sociaux. C'est aussi lors de cette étape que les normes des autres agents sont contrôlés et que des sanctions sont appliquées.

L'ensemble des sous-sections ci-dessous sont des processus effectués à l'intérieur d'une perception qui se caractérise par un ensemble d'agents à observer, une géométrie à l'intérieur de laquelle s'effectue cette perception ainsi que possiblement une condition booléenne. Par exemple, si l'agent A a une perception définie pour l'agent B dans un rayon de 5 mètre autour de lui, les processus définis dans cette perception ne seront appliqués que si l'agent B se trouve à moins de 5 mètres de l'agent A.

Ajout de croyances

Le modélisateur a la possibilité de définir comment **ajouter des croyances** à propos du monde. Pendant cette phase, les informations venant de l'environnement sont transformées en prédicats qui sont ajoutés, soit comme croyance, soit comme incertitude au choix de l'utilisateur, dans les bases cognitives de l'agent. Ce processus permet à l'agent de mettre à jour ses connaissances par rapport à l'environnement, et ainsi de prendre une décision en fonction des derniers états connus du monde.

Du point de vue du modélisateur, cela nécessite de spécifier quelle information venant de l'objet perçu doit être transformée en prédicat ; Une croyance ou une incertitude est créée et ajoutée à la base cognitive correspondante. Par exemple, l'agent A perçoit des feux dans un rayon défini et ajoute comme croyance la position des feux perçus. Si un feu est perçu, le prédicat $feu(["emplacement" :: (x, y)])$ est créé, avec x et y les coordonnées de l'incendie. Par la suite, un état mental cognitif $Belief_{\mathrm{A}}(feu(["emplacement" :: (x, y)]))$ est créé et ajouté à la base des croyances de l'agent A. Le modélisateur peut aussi indiquer une valeur de force particulière ainsi qu'une durée de vie à l'état mental cognitif créé.

L'ajout de croyance est un processus important car il déclenche plusieurs règles. En pratique, l'ajout d'une croyance $Belief_{\mathrm{A}}(\mathrm{X})$ va :

— Retirer la croyance $Belief_{\mathrm{A}}(non\mathrm{X})$.

— Retirer l'intention $Intention_{\mathrm{A}}(\mathrm{X})$.

— Retirer le désir $Desire_{\mathrm{A}}(\mathrm{X})$ si l'intention $Intention_{\mathrm{A}}(\mathrm{X})$ vient d'être retirée.

— Retirer l'incertitude $Uncertainty_{\mathrm{A}}(\mathrm{X})$ ou $Uncertainty_{\mathrm{A}}(non\mathrm{X})$.

— Retirer l'obligation $Obligation_{\mathrm{A}}(\mathrm{X})$.

De même, l'ajout d'une incertitude $Uncertainty_{\mathrm{A}}(\mathrm{X})$ va :

— Retirer l'incertitude $Uncertainty_{\mathrm{A}}(non\mathrm{X})$.

— Retirer la croyance $Belief_{\mathrm{A}}(\mathrm{X})$ ou $Belief_{\mathrm{A}}(non\mathrm{X})$

Contagion émotionnelle

La **contagion émotionnelle** permet de mettre à jour les émotions de l'agent effectuant la perception en fonction des émotions des autres agents perçus, si ces agents possèdent eux-même des émotions. En s'appuyant sur le formalisme détaillé en section 3.2.2, le modélisateur doit indiquer quelle émotion déclenche la contagion, quelle émotion est créée chez l'agent auteur de la perception, ainsi que la valeur de seuil ; les valeurs de charisme et de réceptivité émotionnelle sont automatiquement calculées, comme expliqué en section 4.1.2. La contagion a lieu uniquement si le calcul $charisme \times réceptivité$ est supérieur ou égal au seuil, fixé à 0.25 par défaut.

Si la contagion a lieu, c'est à dire que l'agent perçu possède l'émotion permettant le déclenchement et que les valeurs de charisme et de réceptivités sont suffisantes, l'architecture crée l'émotion indiquée. En s'inspirant du modèle ASCRIBE [BDM+09], l'équation (4.6) indique le calcul déterminant l'intensité de l'émotion créée, dans le cas ou celle-ci existe déjà chez l'agent récepteur ou non, et l'équation (4.7) indique le calcul effectué pour déterminer sa valeur de décroissance. Si l'émotion détectée n'a pas d'intensité (et donc pas de valeur de décroissance), l'émotion créée n'a pas non plus d'intensité (et donc pas de valeur de décroissance).

$$\begin{cases} \text{Si } \mathrm{E}_j(\mathrm{P}) \text{ existe déjà :} & \mathrm{I}[\mathrm{E}_j(\mathrm{P})] = \mathrm{I}[\mathrm{E}_j(\mathrm{P})] + \mathrm{I}[\mathrm{E}_i(\mathrm{P})] \times \mathrm{C}_i \times \mathrm{R}_j \\ \text{sinon :} & \mathrm{I}[\mathrm{E}_j(\mathrm{P})] = \mathrm{I}[\mathrm{E}_i(\mathrm{P})] \times \mathrm{C}_i \times \mathrm{R}_j \end{cases} \tag{4.6}$$

$$\begin{cases} \text{Si } \mathrm{E}_j(\mathrm{P}) \text{ existe déjà :} & \mathrm{D}[\mathrm{E}_j(\mathrm{P})] = \begin{cases} \mathrm{D}[\mathrm{E}_i(\mathrm{P})] \text{ si } \mathrm{I}[\mathrm{E}_i(\mathrm{P})]\, \mathrm{I}[\mathrm{E}_j(\mathrm{P})] \\ \mathrm{D}[\mathrm{E}_j(\mathrm{P})] \text{ si } \mathrm{I}[\mathrm{E}_j(\mathrm{P})]\, \mathrm{I}[\mathrm{E}_i(\mathrm{P})] \end{cases} \\ \text{sinon :} & \mathrm{D}[\mathrm{E}_j(\mathrm{P})] = \mathrm{D}[\mathrm{E}_i(\mathrm{P})] \end{cases} \tag{4.7}$$

L'émotion créée est ajoutée à la base émotionnelle de l'agent réalisant la perception. Les équations (4.6) et (4.7) montrent que, si cette émotion était déjà présente dans la base émotionnelle de l'agent, les deux émotions (celle déjà présente et celle ajoutée) sont unifiées avec pour intensité la somme des deux intensités et pour valeur de décroissance celle liée à l'émotion la plus intense. De cette façon, une émotion avec une forte intensité est plus importante qu'une même émotion avec une intensité plus faible. Par exemple, l'agent A possède une émotion de joie sur un prédicat P, représentée par $Joie_{\mathrm{A}}(\mathrm{P}, 0.5, 0.1)$ et ajoute, par contagion, une émotion de joie sur P représentée par $Joie_{\mathrm{A}}(\mathrm{P}, 0.2, 0.2)$. Après contagion, l'agent A aura dans sa base émotionnelle une seule émotion de joie sur P, représentée par $Joie_{\mathrm{A}}(\mathrm{P}, 0.7, 0.1)$.

Création des relations sociales

Le modélisateur a la possibilité de définir comment **créer des relations sociales** entre l'agent réalisant la perception et les agents perçus. Le modélisateur indique les valeurs initiales de la relation sociale avec l'agent perçu pour chacun des cinq composants de cette relation, comme expliqué en section 3.3.2. Par défaut, si aucune valeur n'est donnée par le modélisateur, une relation neutre est créée, avec chaque composant initialisé à une valeur de 0.0.

Par exemple, le modélisateur définit une création de relation sociale entre les agents A et B, à l'instant où l'agent A perçoit l'agent B, avec chaque composante de la relation initialisée avec une valeur de 0.1. Si cette perception a lieu, la relation sociale qui sera créée et ajoutée à la base des relations sociales de l'agent A sera représentée par $R_{A,B}(0.1, 0.1, 0.1, 0.1, 0.1)$. Il convient de préciser qu'aucune relation sociale n'est créée, à ce moment, par l'agent B envers l'agent A. De plus, si une création de relation sociale de l'agent B envers l'agent A est définie par le modélisateur, celle-ci n'est pas nécessairement avec les mêmes valeurs d'initialisation que la relation de l'agent A envers l'agent B. Enfin, si l'agent A perçoit de nouveau l'agent B, aucune relation sociale ne sera créée puisqu'elle existe déjà.

La seconde façon d'ajouter des relations sociales aux agents est de les créer a priori, au début de la simulation, si l'agent fait partie d'une construction sociale bien définie. Dans ce cas, les relations sociales entre les agents sont définies lors de l'initialisation du modèle et n'ont pas besoin de passer par le module dédié à la création de relations sociales lors de la perception. Pour autant, seules les relations sociales créées dynamiquement, via le module de l'architecture, seront soumises au processus d'évolution automatique expliquée dans la section 4.2.2.

Contrôle des normes

Finalement, l'agent peut **appliquer des sanctions** à travers le contrôle des normes des agents perçus, selon le formalisme défini en section 3.3.1. Le modélisateur indique quelle modalité doit être contrôlée mais aussi quelle sanction et quelle récompense doivent être appliquées. Lors de la perception d'un autre agent soumis au système normatif, l'agent effectuant la perception vérifie si la norme, l'obligation ou la loi contrôlée a été violée, appliquée ou si elle n'était pas activée.

Une norme est considérée comme violée si son contexte est vérifié et que, pour autant, l'agent contrôlé a choisi d'exécuter un autre plan ou une autre norme. Au contraire, une norme est considérée comme appliquée si son contexte est vérifié et que l'agent l'a suivie. Enfin, si le contexte de la norme n'est pas vérifié, celle-ci n'est pas activée et n'est donc pas soumise au contrôle.

Une loi est considérée comme violée si son contexte est vérifié mais que l'agent contrôlé a désobéi, ne créant pas l'obligation qui y est attachée. Si le contexte de la loi contrôlée est vérifié et que l'obligation est créée, la loi est considérée comme appliquée. Enfin, si le contexte de la loi n'est pas vérifié, elle n'est pas activée et n'est donc pas soumise au contrôle.

Une obligation est considérée comme violée si elle se trouve dans la base des obligations de l'agent contrôlé mais que celui-ci a choisi d'exécuter un plan, ou une norme, qui ne répond pas à cette obligation. Une obligation est considérée comme appliquée si elle se trouve dans la base des obligations de l'agent contrôlé et que celui-ci exécute une norme répondant à cette obligation. Finalement, si l'obligation ne se trouve pas dans la base des obligations de l'agent contrôlé, ni la sanction, ni la récompense n'est exécutée.

Contrairement aux normes et aux lois, les sanctions ou les récompenses ne sont appliquées qu'une seule fois lors du contrôle d'une obligation. Cela signifie qu'une fois contrôlée, une obligation ne peut être de nouveau sanctionnée ou récompensée, tant que l'agent contrôlé n'a pas reconsidéré ses plans et normes, et n'a pas de nouveau possiblement désobéi à son obligation. Cela permet de faire une différence dans le traitement des lois et des obligations, et donc d'offrir le plus d'options possibles aux utilisateurs.

Les sanctions et les récompenses sont des comportements exécutés par l'agent effectuant le contrôle, pouvant agir sur l'agent contrôlé ou sur l'agent effectuant le contrôle. Lors de la définition d'un contrôle, le modélisateur doit définir au moins une sanction ou une récompense. En

revanche, il n'est pas nécessaire de définir une sanction et une récompense par contrôle.

Par exemple, un agent A possède la norme $norme1_A(P1, C, 0.5, 4, Comp1, 1)$ qui répond à une intention portant sur le prédicat P1, dans un contexte C, avec une valeur d'obéissance de 0.5, une priorité de 4, réalisant un comportement connu sous le nom de $Comp1$ et avec une durée de violation correspondant à une étape de calcul. Un agent B pourra effectuer sur l'agent A le contrôle $(norme1_A, Sanc1, Recomp1)$ et exécutera la sanction nommée $Sanc1$ si la norme contrôlée est violée, ou la récompense $Recomp1$ si la norme contrôlée est respectée.

Si l'agent A possède une loi $loi1(C, P1, 0.5)$ qui crée une obligation portant sur le prédicat P1 dans le contexte C si la valeur d'obéissance de l'agent est inférieur à 0.5, l'agent B peut effectuer un contrôle $(loi1, Sanc2, Recomp2)$ qui exécutera la sanction $Sanc2$ si la loi est violée ou qui exécutera la récompense $Recomp2$ si la loi a été suivie. L'agent B pourrait aussi disposer d'un contrôle $(P1, Sanc3, Recomp3)$ qui vérifiera s'il existe une obligation portant sur le prédicat P1 dans la base des obligations de l'agent A, puis si l'action en cours de l'agent A est une norme permettant de répondre à une intention portant sur le prédicat P1. Si c'est le cas, la récompense $Recomp3$ sera exécutée et si c'est une autre norme ou un autre plan que l'agent A est en train d'exécuter, c'est la sanction $Sanc3$ qui sera exécutée.

4.2.2 Module de mise à jour des connaissances

La deuxième étape de l'architecture, correspondant au module numéro 2 sur la figure 4.2, permet de gérer les connaissances de l'agent avant la prise de décision. Dans ce module, les bases de connaissances sont mises à jour en fonction du module des perceptions, en ajoutant des états mentaux cognitifs, des obligations, des émotions et en faisant évoluer les relations sociales. Chacun des processus du deuxième module sont détaillés dans cette section.

Règles d'inférence

Pour gérer les états mentaux cognitifs en fonction des dernières perceptions, le modélisateur peut définir des **règles d'inférence**. Ces règles sont déclenchées par une croyance, une incertitude ou une émotion nouvellement ajoutée, dans un contexte booléen particulier, et permettent d'ajouter ou de retirer n'importe quel état mental cognitif ou n'importe qu'elle émotion indiquée par l'utilisateur. La définition de plusieurs règles d'inférences permet à l'agent d'adapter ses connaissances à la situation perçue sans pour autant effacer les précédents états mentaux cognitifs, permettant ainsi de créer un comportement cognitif.

Une règle d'inférence peut être vue comme une règle "si-alors" : par exemple, le modélisateur peut définir une règle qui indique que, si l'agent possède une croyance $Belief_i(feu)$, alors il acquiert le désir $Desire_i(fuir)$ et retire son intention $Intention_i(se\_promener)$. Cette règle signifie que si l'agent possède une croyance relative à un feu, il perd son intention de se promener et aura le désir de fuir. Cette règle peut être complétée par un contexte particulier indiquant qu'elle n'est pas activée si l'agent est déjà en train de fuir. De la même façon, il est possible de définir plusieurs états cognitifs mentaux comme éléments déclencheurs, comme éléments créés ou comme éléments retirés.

Définition de lois

En utilisant le même modèle que les règles d'inférence, le modélisateur peut définir des **lois**, en s'appuyant sur le formalisme définit en section 3.3.1. Ces lois permettent de créer des obligations dans un certain contexte, en fonction des croyances créées à travers la perception ou par les règles d'inférence. Le modélisateur renseigne un seuil d'obéissance à la loi qui, s'il n'est pas dépassé par la valeur d'obéissance de l'agent, indique que la loi est violée. Si la loi est activée, elle crée une obligation qui est ajoutée à la base des obligations de l'agent. En définissant des lois, le modélisateur peut alors définir un comportement basé sur les obligations pour l'agent simulé.

Pour donner un exemple, une loi peut se définir comme une règle "si-alors" où, dans un contexte C particulier évalué de façon booléenne, une obligation est ajoutée sur un prédicat comme

par exemple $Obligation_i(fuir)$ qui représente l'obligation de fuir. Une valeur d'obéissance inférieure à 1, par exemple 0.8, ajoutée à la loi permet d'offrir la possibilité aux agents de ne pas la suivre, ce qui donne, selon le formalisme détaillé en section 3.3.1, la loi $loi1(C, fuir, 0.8)$.

Moteur émotionnel

BEN permet aux agents de générer des émotions à partir des états mentaux cognitifs. Cette **création d'émotion** s'appuie sur le modèle OCC [OCC90] et sur le formalisme logique proposé par Adam [Ada07], qui propose de représenter le modèle émotionnel via les concepts du modèle BDI.

D'après la théorie OCC, les émotions peuvent être réparties selon trois groupes : les émotions liées aux événements, les émotions liées aux personnes et aux actions effectuées par ces personnes, et les émotions liées aux objets. Dans BEN, le comportement des agents est défini par rapport à un contexte social donc seules les deux premières catégories d'émotions (les émotions liées aux événements et les émotions liées aux personnes et à leurs actions) sont prises en considération.

Les vingt émotions définies dans l'architecture BEN peuvent être divisées en sept groupes en fonction de leurs relations avec les états mentaux en jeu : les émotions à propos des croyances, les émotions à propos des incertitudes, les émotions combinées à propos des incertitudes, les émotions à propos d'autres agents avec une valeur d'appréciation positive, les émotions à propos d'autres agents avec une valeur d'appréciation négative, les émotions à propos des idéaux, et les émotions combinées à propos des idéaux. Toutes les intensités et valeurs initiales de décroissance sont calculées en utilisant le modèle OCEAN de la personnalité [MJ92] défini en section 3.2.1 ainsi que les valeurs attachées aux états mentaux cognitifs en jeu, développés en section 3.1.2.

Les émotions à propos des croyances sont la joie et la tristesse et s'expriment de la façon suivante :

— $\mathbf{Joy}_i(\mathbf{P}_j,\mathbf{j}) \stackrel{\text{def}}{=} \text{Belief}_i(\text{P}_j)$ & $\text{Desire}_i(\text{P})$

— $\mathbf{Sadness}_i(\mathbf{P}_j,\mathbf{j}) \stackrel{\text{def}}{=} \text{Belief}_i(\text{P}_j)$ & $\text{Desire}_i(\text{not P})$

Leur intensité initiale est calculée selon l'équation (4.8) avec N la composante neurotique du modèle OCEAN.

$$\text{I}[\text{E}_i(\text{P})] = \text{V}[\text{B}elief_i(\text{P})] \times \text{V}[\text{D}esire_i(\text{P})] \times (1 + (0,5 - \text{N})) \tag{4.8}$$

Les émotions à propos des incertitudes sont la peur et l'espoir et sont définies de la façon suivante :

— $\mathbf{Hope}_i(\mathbf{P}_j,\mathbf{j}) \stackrel{\text{def}}{=} \text{Uncertainty}_i(\text{P}_j)$ & $\text{Desire}_i(\text{P})$

— $\mathbf{Fear}_i(\mathbf{P}_j,\mathbf{j}) \stackrel{\text{def}}{=} \text{Uncertainty}_i(\text{P}_j)$ & $\text{Desire}_i(\text{not P})$

Leur intensité initiale est calculée selon l'équation (4.9).

$$\text{I}[\text{E}_i(\text{P})] = \text{V}[\text{U}ncertainty_i(\text{P})] \times \text{V}[\text{D}esire_i(\text{P})] \times (1 + (0,5 - \text{N})) \tag{4.9}$$

Les émotions combinées à propos des incertitudes sont des émotions créées à partir de la peur de de l'espoir. Elles apparaissent lorsqu'une incertitude devient une croyance, transformant la peur et l'espoir en satisfaction, déception, soulagement et peur confirmée. Ces émotions sont définies de la façon suivante :

— $\mathbf{Satisfaction}_i(\mathbf{P}_j,\mathbf{j}) \stackrel{\text{def}}{=} \text{Hope}_i(\text{P}_j,\text{j})$ & $\text{Belief}_i(\text{P}_j)$

— $\mathbf{Disappointment}_i(\mathbf{P}_j,\mathbf{j}) \stackrel{\text{def}}{=} \text{Hope}_i(\text{P}_j,\text{j})$ & $\text{Belief}_i(\text{not P}_j)$

— $\mathbf{Relief}_i(\mathbf{P}_j,\mathbf{j}) \stackrel{\text{def}}{=} \text{Fear}_i(\text{P}_j,\text{j})$ & $\text{Belief}_i(\text{not P}_j)$

— $\mathbf{Fear\ Confirmed}_i(\mathbf{P}_j,\mathbf{j}) \stackrel{\text{def}}{=} \text{Fear}_i(\text{P}_j,\text{j})$ & $\text{Belief}_i(\text{P}_j)$

Leur intensité initiale est calculée en fonction de l'équation (4.10) avec E'$_i$(P) l'émotion de peur ou d'espoir.

$$I[E_i(P)] = V[Belief_i(P)] \times I[E'_i(P)] \tag{4.10}$$

En plus de ces définitions, le formalisme logique [Ada07] identifie quatre règles d'inférences déclenchées par l'apparition de ces émotions :

— La création de **peur confirmée** ou la création de **soulagement** prend la place de l'émotion de **peur**.

— La création de **satisfaction** ou la création de **déception** prend la place de l'émotion d'**espoir**.

— La création de **satisfaction** ou de **soulagement** implique la création de **joie**.

— La création de **déception** ou de **peur confirmée** implique la création de **tristesse**.

Les émotions à propos d'autres agents avec une appréciation positive sont des émotions liées aux émotions des autres agents avec qui une relation sociale, dont la valeur d'appréciation est positive, existe dans la base des relations sociales. Ces émotions sont appelées "content pour" et "désolé pour" et se définissent de la façon suivante :

— **Happy For**$_i$**(P,j)** $\stackrel{\text{def}}{=}$ L[R$_{i,j}$] > 0 & Belief$_i$(Joy$_j$(P))

— **Sorry For**$_i$**(P,j)** $\stackrel{\text{def}}{=}$ L[R$_{i,j}$] > 0 & Belief$_i$(Sadness$_j$(P))

Leur intensité initiale est calculée selon l'équation (4.11) avec A la valeur d'agréabilité du modèle OCEAN.

$$I[E_i(P)] = I[E_j(P)] \times L[R_{i,j}] \times (1 - (0,5 - A)) \tag{4.11}$$

Les émotions à propos des autres agents avec une appréciation négative sont proches de la définition précédente. Cependant, elles sont liées aux émotions des autres agents avec qui une relation sociale, dont la valeur d'appréciation est négative, existe dans la base des relations sociales. Ces émotions sont le ressentiment et la jubilation et se définissent de la façon suivante :

— **Resentment**$_i$**(P,j)** $\stackrel{\text{def}}{=}$ L[R$_{i,j}$] < 0 & Belief$_i$(Joy$_j$(P))

— **Gloating**$_i$**(P,j)** $\stackrel{\text{def}}{=}$ L[R$_{i,j}$)] < 0 & Belief$_i$(Sadness$_j$(P))

Leur intensité initiale est calculée selon l'équation (4.12) qui peut être vue comme l'inverse de l'équation (4.11). Elle signifie que l'intensité du ressentiment ou de la jubilation est d'autant plus grande que l'agent possède une faible valeur d'agréabilité, ce qui est le contraire pour les intensités des émotions "content pour" et "désolé pour".

$$I[E_i(P)] = I[E_j(P)] \times |L[R_{i,j}]| \times (1 + (0,5 - A)) \tag{4.12}$$

Les émotions à propos des idéaux sont liées à la base des idéaux de l'agent qui contient, au lancement de la simulation, l'ensemble des prédicats auxquels l'agent a donné une valeur morale. Ces idéaux peuvent être louables (leur valeur morale est positive) ou blâmables (leur valeur morale est négative). Les émotions venant des idéaux sont la fierté, la honte, l'admiration et le reproche et ont la définition suivante :

— **Pride**$_i$**(P$_i$,i)** $\stackrel{\text{def}}{=}$ Belief$_i$(P$_i$) & Ideal$_i$(P$_i$) & V(Ideal$_i$(P$_i$)) > 0

— **Shame**$_i$**(P$_i$,i)** $\stackrel{\text{def}}{=}$ Belief$_i$(P$_i$) & Ideal$_i$(P$_i$) & V(Ideal$_i$(P$_i$)) < 0

— **Admiration**$_i$**(P$_j$,j)** $\stackrel{\text{def}}{=}$ Belief$_i$(P$_j$) & Ideal$_i$(P$_i$) & V(Ideal$_i$(P$_i$)) > 0

— **Reproache**$_i$**(P$_j$,j)** $\stackrel{\text{def}}{=}$ Belief$_i$(P$_j$) & Ideal$_i$(P$_i$) & V(Ideal$_i$(P$_i$)) < 0

Leur intensité initiale est calculée selon l'équation (4.13) avec O la composante d'ouverture du modèle OCEAN.

$$\mathrm{I}[\mathrm{E}_i(\mathrm{P})] = \mathrm{V}[\mathrm{B}elief_i(\mathrm{P})] \times |\mathrm{V}[\mathrm{I}deal_i(\mathrm{P})]| \times (1 + (0{,}5 - \mathrm{O})) \tag{4.13}$$

Finalement, les émotions combinées à propos des idéaux sont des émotions créées à partir de la fierté, de la honte, de l'admiration et du reproche. Elles apparaissent lorsque de la joie ou de la tristesse sont créées en lien avec une émotion sur un idéal. Ces émotions sont la gratification, le remord, la gratitude et la colère et sont définies de la façon suivante :

— **Gratification**$_i$**(P**$_i$**,i)** $\stackrel{\text{def}}{=}$ Pride$_i$(P$_i$,i) & Joy$_i$(P$_i$)

— **Remorse**$_i$**(P**$_i$**,i)** $\stackrel{\text{def}}{=}$ Shame$_i$(P$_i$,i) & Sadness$_i$(P$_i$)

— **Gratitude**$_i$**(P**$_j$**,j)** $\stackrel{\text{def}}{=}$ Admiration$_i$(P$_j$,j) & Joy$_i$(P$_j$)

— **Anger**$_i$**(P**$_j$**,j)** $\stackrel{\text{def}}{=}$ Reproache$_i$(P$_j$,j) & Sadness$_i$(P$_i$)

Leur intensité initiale est calculée selon l'équation (4.14) avec E'$_i$(P) l'émotion à propos de l'idéal et E"$_i$(P) l'émotion à propos de la croyance.

$$\mathrm{I}[\mathrm{E}_i(\mathrm{P})] = \mathrm{I}[\mathrm{E}'_i(\mathrm{P})] \times \mathrm{I}[\mathrm{E}"_i(\mathrm{P})] \tag{4.14}$$

Pour conserver ces intensités entre 0 et 1, les équations (4.8), (4.9), (4.10), (4.11), (4.12), (4.13) et (4.14) sont tronquées entre 0 et 1 si nécessaire.

La valeur initiale de décroissance pour chacune des vingt émotions est calculée en utilisant l'équation (4.15) avec Δ t un pas de temps permettant de définir la durée de vie moyenne d'une émotion, initialisée à une heure par défaut.

$$\mathrm{D}[\mathrm{E}_i(\mathrm{P})] = \mathrm{N} \times \mathrm{I}[\mathrm{E}_i(\mathrm{P})] \times \Delta t \tag{4.15}$$

Du point de vue du modélisateur, l'ensemble de ces règles sont automatiques. Cela signifie que si le processus de création des émotions est activé, l'agent va exécuter les règles de création d'émotions en fonction des modifications des états mentaux cognitifs de l'agent ; l'ajout de croyances ou d'incertitudes va déclencher les règles contenant ces états mentaux cognitifs. Seules les émotions "content pour", "désolé pour", ressentiment et jubilation sont calculées à chaque pas de temps.

Si une émotion créée automatiquement par le moteur émotionnel était déjà présente dans la base des émotions de l'agent (par exemple, l'agent A possède une émotion Joy_{A}(P) et le moteur émotionnel crée une nouvelle émotion Joy_{A}(P)), ces deux émotions vont être unifiées en additionnant leur intensité et en conservant la valeur de décroissance de l'émotion ayant la plus forte intensité.

Enfin, si le modélisateur n'utilise pas la personnalité, le moteur émotionnel va tout de même fonctionner et créer des émotions. Pour autant, ces émotions n'auront pas d'intensité initiale ni de valeur de décroissance initiale, ces valeurs pourront alors être renseignées manuellement par l'utilisateur. Il en est de même si le modélisateur n'utilise pas les relations sociales, les émotions "content pour", "désolé pour", ressentiment et jubilation ne seront pas créées par le moteur émotionnel de BEN mais les autres émotions pourront être créées.

Mise à jour sociale

Lorsqu'un agent déjà connu est de nouveau perçu (c.a.d il existe déjà une relation sociale avec lui), la relation avec cet agent est mise à jour automatiquement par l'architecture BEN. Cette mise à jour s'appuie sur le travail de Ochs [OSC09b] et prend en compte les états mentaux cognitifs de l'agent ainsi que ses émotions. Dans cette sous-section, la mise à jour automatique par l'architecture d'un lien social $\mathrm{R}_{i,j}$(L, D, S, F) est détaillée ; pour autant, la valeur de confiance de ce lien n'est pas mise à jour automatiquement.

— **Appréciation** : Selon Ortony [OKC91], le degré d'appréciation entre deux agents dépend de la valence (positive ou négative) de l'émotion créée par l'agent envers qui est tournée la relation sociale. Dans le modèle émotionnel développé dans l'architecture BEN, la *joie* et l'*espoir* sont considérées comme des émotions positives (la *satisfaction* et le *soulagement* créent automatiquement de la *joie*) alors que la *tristesse* et la *peur* sont considérées comme des émotions négatives (la *peur confirmée* et la *déception* créent automatiquement de la *tristesse*). Ainsi, si un agent i ressent une émotion positive (respectivement négative) causée par l'agent j, cela va augmenter (respectivement diminuer) la valeur d'appréciation du lien social entre i et j.

  De plus, des travaux ont montré que le degré d'appréciation est influencé par la valeur de solidarité entre deux agents [SMC14]. Cela s'explique par le fait que les gens ont tendance à apprécier les personnes avec qui elles partagent des similarités.

  La formule de calcul d'évolution de la valeur d'appréciation est décrite par l'équation (4.16) avec mPos la moyenne des intensités des émotions positives causées par l'agent j, mNeg la moyenne des intensités des émotions négatives causées par l'agent j et α_{L} un coefficient dépendant de la personnalité de l'agent, indiquant l'importance des émotions dans le processus et qui est décrit par l'équation (4.17).

$$\mathrm{L}[\mathrm{R}_{i,j}] = \mathrm{L}[\mathrm{R}_{i,j}] + |\mathrm{L}[\mathrm{R}_{i,j}]|(1 - |\mathrm{L}[\mathrm{R}_{i,j}]|)\mathrm{S}[\mathrm{R}_{i,j}] + \alpha_{\mathrm{L}}(1 - |\mathrm{L}[\mathrm{R}_{i,j}]|)(m\mathrm{P}os - m\mathrm{N}eg) \tag{4.16}$$

$$\alpha_{\mathrm{L}} = 1 - \mathrm{N} \tag{4.17}$$

— **Dominance** : Keltner et Haid [KH01] et Shiota *et al.* [SCKH04] expliquent que l'émotion de *peur* ou de *tristesse* causée par un autre agent représente un statut inférieur. Dans le même temps, Knutson [Knu96] explique que la perception de *peur* et de *tristesse* chez les autres augmente la sensation de pouvoir sur ces autres personnes.

  La formule de calcul de l'évolution de la dominance est décrite par l'équation (4.18) avec mSE la moyenne des intensités des émotions négatives causées par l'agent i chez l'agent j, mOE la moyenne des intensités des émotions négatives causées par l'agent j chez l'agent i et α_{D} un coefficient dépendant de la personnalité de l'agent, indiquant l'importance des émotions dans le processus et qui est décrit par l'équation (4.19).

$$\mathrm{D}[\mathrm{R}_{i,j}] = \mathrm{D}[\mathrm{R}_{i,j}] + \alpha_{\mathrm{D}}(1 - |\mathrm{D}[\mathrm{R}_{i,j}]|)(m\mathrm{SE} - m\mathrm{OE}) \tag{4.18}$$

$$\alpha_{\mathrm{D}} = 1 - \mathrm{N} \tag{4.19}$$

— **Solidarité** : Comme expliqué dans le formalisme détaillé en section 3.3.2, la solidarité représente le degré de similarité des désirs, des croyances et des incertitudes entre deux agents. Dans BEN, l'évolution de la solidarité dépend du ratio de similarité entre les désirs, les croyances et les incertitudes de l'agent i et ceux de l'agent j. Pour calculer ces similarités et ces oppositions, l'agent i a besoin d'avoir des croyances sur les états mentaux cognitifs de l'agent j pour ensuite comparer ces connaissances avec ses propres états mentaux cognitifs, afin de détecter des similarités et des oppositions dans ces connaissances.

  De plus, selon de Rivera et Grinkis [dRG86], les émotions négatives ont tendances à diminuer la valeur de solidarité entre deux personnes. La formule de calcul de l'évolution de la solidarité est décrite par l'équation (4.20) avec sim le nombre d'états mentaux cognitifs similaires entres les agents i et j, opp le nombre d'états mentaux cognitifs opposés entre les agents i et j, NbKnow le nombre d'état mentaux cognitifs en commun entre les agents i et j, mNeg la moyenne des intensités des émotions négatives causées par l'agent j, α_{S1} un coefficient dépendant de la personnalité de l'agent, indiquant l'importance des similarités et des oppositions dans le processus, décrit par l'équation (4.21) et α_{S2} un coefficient dépendant

de la personnalité de l'agent, indiquant l'importance des émotions dans le processus, décrit par l'équation (4.22).

$$S[R_{i,j}] = S[R_{i,j}] + S[R_{i,j}] \times (1 - S[R_{i,j}]) \times (\alpha_{S1} \frac{sim - opp}{NbKnow} - \alpha_{S2} mNeg)) \tag{4.20}$$

$$\alpha_{S1} = 1 - O \tag{4.21}$$

$$\alpha_{S2} = 1 - N \tag{4.22}$$

— **Familiarité** : En psychologie, les émotions et la cognition ne semblent pas avoir un impact sur l'évolution de la familiarité. Cependant, Collins et Miller [CM94] expliquent que les gens ont tendance à être plus familier avec les personnes qu'ils apprécient. Cette notion est modélisée en appuyant l'évolution de la valeur de familiarité sur la valeur d'appréciation entre les deux agents. La formule de calcul est définie par l'équation (4.23).

$$F[R_{i,j}] = F[R_{i,j}](1 + L[R_{i,j}]) \tag{4.23}$$

Toutes les équations d'évolutions ont été élaborées de telle sorte que les valeurs d'appréciation et de dominance restent entre -1 et 1 mais aussi pour que les valeurs de solidarité et de familiarité restent entre 0 et 1, en accord avec le formalisme détaillé en section 3.3.2.

La valeur de confiance n'évolue pas automatiquement dans BEN car il n'y a pas de lien clair et automatique avec les émotions et la cognition. Cependant, cette valeur peut être modifiée, en particulier à travers la définition de sanctions et de récompenses à des normes sociales où le modélisateur peut indiquer une modification de la valeur de confiance pendant le processus de contrôle décrit en section 4.2.1.

4.2.3 Moteur cognitif et normatif

La troisième partie de l'architecture, numérotée 3 sur la figure 4.2, est la seule partie obligatoire de BEN. C'est dans ce module que l'agent prend une décision et exécute une action à la suite de la décision prise, à travers un moteur cognitif. De façon optionnelle, un moteur normatif peut être utilisé pour ajouter un système normatif au moteur cognitif, ou pour ne prendre une décision que sur un système normatif. Une fois le raisonnement effectué, l'agent possède une intention qui amène au choix d'un plan d'action ou d'une norme, qui sera ensuite exécutée, modifiant l'environnement.

Dans cette section, le fonctionnement des moteurs cognitifs et normatifs, permettant à l'agent de raisonner et de prendre une décision, est décrit en détail.

Moteur cognitif

La prise de décision cognitive se fait, dans BEN, en s'appuyant sur le modèle BDI [Bra87], particulièrement sur son implémentation dans la plateforme GAMA [TBC+16]. Dans cette implémentation du modèle BDI, l'agent possède une base des désirs, une base des intentions, une base de plans d'action, une intention courante et un plan courant. Ainsi, les intentions sont définies en fonction des désirs, avec la possibilité de définir des sous-intentions. L'intention courante est définie comme la dernière intention ajoutée à la base dont il existe au moins un plan permettant d'y répondre dans le contexte actuel. Cette intention courante va être conservée jusqu'à ce qu'elle soit satisfaite, c'est à dire que la croyance qu'elle a été satisfaite a été ajoutée à la base des croyances de l'agent à la fin d'un plan ou dans une perception, comme expliqué en section 4.2.1. Le plan courant est le plan sélectionné pour répondre à l'intention courante.

Le moteur cognitif, processus permettant à l'agent de prendre une décision en fonction de ses états mentaux cognitifs et de son contexte, est décrit par la figure 4.3.

Le processus de prise de décision cognitif peut être divisé en cinq étapes :

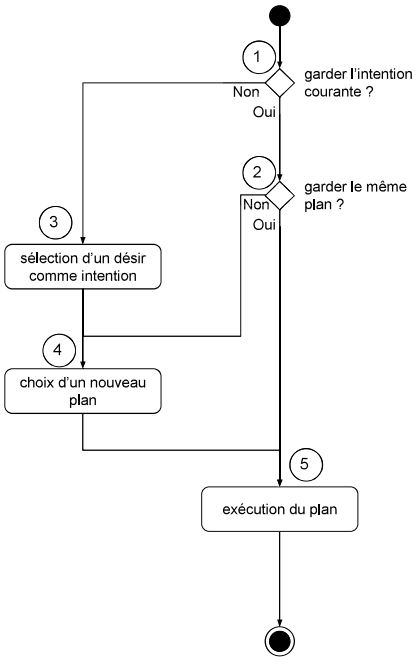

FIGURE 4.3 – Diagramme représentant le moteur cognitif de l'architecture BEN

— **Étape 1** : le moteur vérifie l'intention courante. Si celle-ci est toujours valide (c.a.d il existe toujours un plan activable dans le contexte actuelle répondant à l'intention courante), elle est conservée et l'agent persévère dans son intention. Par contre, si celle-ci n'est plus valide, qu'elle est remise en cause par la probabilité de remise en cause discutée en section 4.1.2, ou qu'il n'y a plus d'intention courante, l'agent raisonne sur ses désirs pour acquérir une nouvelle intention courante.
— **Étape 2** : le moteur vérifie si le plan en cours est toujours utilisable ou non, en fonction du contexte mis à jour par la perception. Si le contexte est toujours valide, l'agent persévère dans son action. Sinon, il choisit un nouveau plan. Le plan en cours peut aussi être remis en cause de façon aléatoire, en fonction de la probabilité de remise en cause du plan discutée en section 4.1.2, forçant l'agent à choisir un nouveau plan courant.
— **Étape 3** : si la base des intentions est vide, le désir de plus haute priorité, dont il existe un plan pouvant y répondre dans le contexte actuel, est ajouté à la base des intentions et sélectionné comme intention courante. Si les désirs ont tous la même priorité ou aucune priorité, le choix est aléatoire. Si la base des intentions n'est pas vide, l'intention ajoutée en dernier devient l'intention courante.
— **Étape 4** : le plan de plus haute priorité répondant à l'intention courante et dont le contexte est valide est sélectionné comme plan en cours.
— **Étape 5** : le plan courant est exécuté. Ici, le plan représente une unité atomique d'action de l'agent, celui-ci est donc exécuté en entier à cette étape.

Les étapes 3 et 4 ne sont pas nécessairement déterministes ; elles peuvent être probabilistes. Dans ce cas, les valeurs de priorité des désirs et des plans servent de probabilités. Par exemple, si un désir A possède une priorité de 2 et un désir B possède une priorité de 1, le désir A aura 2 chances sur 3 d'être sélectionné par un agent, quand le désir B aura 1 chance sur 3 d'être sélectionné.

Le formalisme des plans, détaillé en section 3.1.2, indique qu'ils sont dépendants d'un contexte. Ce contexte est évalué de façon booléenne et peut contenir aussi bien des informations extérieures

à l'agent que des informations sur les états mentaux cognitifs, les émotions ou les relations sociales de ce dernier. Ainsi, les dimensions affectives et sociales agissent sur la cognition, en rendant valides ou invalides certains plans d'actions.

Si l'utilisateur n'utilise aucun des processus optionnels de BEN (ni la perception, ni la gestion des connaissances), le moteur cognitif continue de fonctionner. Seulement, le modélisateur sera obligé de gérer manuellement les états mentaux cognitifs. Cela signifie ajouter ou supprimer des connaissances pour faire évoluer les bases des désirs et des intentions, et ainsi permettre à l'agent de changer de décision en fonction des changements de l'environnement. Enfin, s'il n'y a aucun désir ou aucun plan activable dans le contexte actuel, l'agent ne fait rien et prendra une décision d'action plus tard, lorsque le contexte ou sa base des désirs auront changés.

Moteur normatif

La prise de décision en fonction des obligations et des normes se fait par le biais d'un moteur normatif qui se place en parallèle du moteur cognitif, sur le modèle de l'architecture BOID [BDH+01]. Ce moteur est activé à partir du moment où l'agent possède des obligations ou des normes, selon le formalisme présenté en section 3.3.1.

Le moteur cognitif et normatif présent dans BEN, permettant à un agent de prendre une décision en fonction de ses états mentaux cognitifs et de ses éléments normatifs, est décrit par la figure 4.4.

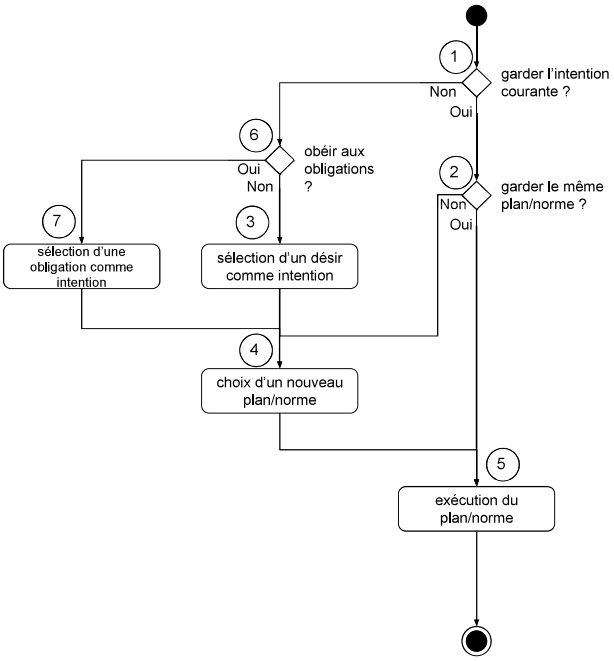

FIGURE 4.4 – Diagramme représentant le moteur cognitif et normatif de l'architecture BEN

Sur cette figure, les étapes 1, 2, 3, 4 et 5 correspondent au moteur cognitif présenté en section 4.2.3. Cela signifie que le moteur normatif agit en supplément du moteur cognitif et non en remplacement de ce dernier. Les étapes Les nouvelles étapes de raisonnement ajoutée par le moteur normatif sont les suivantes :

— **Étape 6** : le moteur vérifie si l'agent obéit à une obligation, parmi les obligations dont il existe une norme permettant d'y répondre dans le contexte actuel et ayant un seuil d'obéissance inférieur à la valeur d'obéissance de l'agent. Si l'agent désobéit à ses obligations, la prise de décision se fera en fonction des désirs.

— **Étape 7** : l'obligation avec la plus haute priorité, dont il existe une norme permettant d'y répondre dans le contexte actuel, est sélectionnée pour devenir l'intention courante.

Comme pour le moteur cognitif, l'étape 7 du moteur normatif n'est pas nécessairement déterministe ; elle peut être probabiliste. Dans ce cas, les valeurs de priorité des obligations servent de probabilités d'être sélectionné.

Dans l'étape 4, les normes de l'agent sont évaluées en priorité sur les plans. Cela signifie que le moteur va d'abord passer en revue toutes les normes activables dans le contexte actuel et répondant à l'intention en cours. Si une de ces normes possède une valeur d'obéissance inférieure à la valeur d'obéissance de l'agent, celle-ci sera sélectionnée comme norme courante et exécutée dans l'étape 7. Par contre, s'il n'existe pas de telle norme, le moteur de raisonnement va alors considérer les plans d'actions répondant à l'intention en cours et activables dans le contexte actuel pour en sélectionner un comme plan courant qui sera alors exécuté dans l'étape 7.

S'il n'y a pas d'obligations ou de normes, ponctuellement, le moteur normatif fonctionne simplement comme le moteur cognitif. À l'opposé, s'il n'y a pas de désirs ou de plans d'action, le moteur normatif prend le pas sur le moteur cognitif. Dans ce dernier cas, le comportement de l'agent n'est géré que par sa dimension normative, rendant la dimension cognitive de l'architecture optionnelle.

4.2.4 Dégradations temporelles

La dernière partie de l'architecture BEN, numérotée 4 sur la figure 4.2, permet de créer une dynamique temporelle dans le comportement de l'agent. Pour ce faire, ce module dégrade automatiquement les états mentaux cognitifs et les émotions puis met à jour le statut des normes. Ces différents processus sont détaillés dans cette section.

Dégradation des prédicats

La **dégradation des états mentaux cognitifs** consiste en la réduction de la valeur de durée de vie de chacun des états mentaux cognitifs stockés par l'agent. Lorsque la valeur de durée de vie devient nulle, l'état mental cognitif qui y est lié est retiré des bases de l'agent. Ce processus modélise un mécanisme d'oubli, où les croyances, incertitudes, désirs et idéaux peuvent être oubliés par l'agent après un certain temps.

Prenons l'exemple d'une croyance $Belief_i(\mathrm{P}, 10)$ portant sur un prédicat P et avec une durée de vie de valeur 10. Une fois que l'agent aura pris sa décision concernant l'intention courante et le plan d'action à exécuter, la croyance de l'agent est dégradée. Ainsi, lors du prochain pas de calcul dans lequel l'agent devra prendre une décision, sa croyance sera devenue $Belief_i(\mathrm{P}, 9)$. Après 9 autre étapes de calcul, la durée de vie de cette croyance atteindra 0 et la croyance sera retirée de la base des croyances de l'agent. Par contre, si la croyance $Belief_i(\mathrm{P}, n)$ est ajoutée alors que la précédente croyance n'avait pas été oubliée, cette dernière croyance remplacera la précédente, quelque soit la valeur n de la durée de vie.

Comme ce processus est optionnel, il est possible de ne pas dégrader la durée de vie des états mentaux cognitifs. Cela revient à créer un état mental dont la durée de vie est à l'infini. Le modélisateur peut utiliser en même temps, dans l'architecture BEN, des états mentaux cognitifs avec une durée de vie finie (qui seront donc dégradés à chaque pas de calcul) et des états mentaux cognitifs à durée de vie infinie.

Dégradation des émotions

La **dégradation des émotions** consiste en la réduction de l'intensité de chaque émotion présente dans la base des émotions de l'agent, en fonction des valeurs de décroissance de chaque émotion. Lorsque les intensités sont nulles (ou la première fois qu'elles sont inférieures à zéro), elles sont retirées de la base émotionnelle de l'agent. Encore une fois, ce processus permet de modéliser un mécanisme d'oubli lié aux émotions. Ainsi, une émotion n'existe que pendant un

certain temps, quelque soit son intensité ; une émotion avec une forte intensité peut n'avoir un impact que limité dans le temps avec une forte valeur de décroissance, face à une émotion à l'intensité plus faible qui pourrait prendre de l'importance dans le temps avec une faible valeur de décroissance.

Prenons par exemple l'émotion $Joie_i(P, 5.0, 2.0)$ qui est ressentie à un agent i. Après la phase de raisonnement qui a permis à l'agent de choisir une action à effectuer en fonction de son contexte, l'émotion est dégradée en soustrayant à l'intensité la valeur de décroissance. L'émotion devient alors $Joie_i(P, 3.0, 2.0)$. Deux tours plus tard, l'émotion aura une intensité, après dégradation, de valeur -1.0 et sera retirée de la base des émotions de l'agent ; elle ne sera plus ressentie par l'agent. Par contre, si une émotion $Joie_i(P, i, d)$ est ajoutée alors que la précédente est toujours ressentie par l'agent, ce dernier aura alors une émotion de joie à propos d'un prédicat p avec pour intensité $5.0 + i$ et pour valeur de décroissance celle attachée à l'émotion ayant l'intensité maximale entre 5.0 et i.

Comme ce processus est optionnel, il est possible de ne pas dégrader l'état émotionnel de l'agent automatiquement. Cela arrive quand l'émotion n'a pas de valeur de décroissance précisée. Dans ce cas, l'intensité émotionnelle n'évolue pas de façon automatique dans l'architecture, et c'est au modélisateur d'inclure un système de dégradation temporelle manuellement. Il est possible, dans BEN, d'utiliser en même temps des émotions possédant une valeur de décroissance et des émotions n'en possédant pas.

Mise à jour des normes

Finalement, la **mise à jour de la violation des normes** indique si les normes de l'agent étaient activables ou non (si leur contexte était valide ou non) et si elles ont été violées ou non (la norme était activable mais l'agent y a désobéi). Une norme est alors violée pour un certain temps spécifié par le modélisateur. Cette durée de vie de la violation est mise à jour et si elle tombe à zéro, la norme n'est plus considérée comme violée et ne peut donc plus être sanctionnée lors du contrôle d'un autre agent, jusqu'à ce qu'elle soit de nouveau violée.

Prenons le cas d'une norme $norme1_i(P, C, 0.5, 2.0, Comp1, 3)$ répondant à une intention sur le prédicat P, dans un contexte C, avec pour valeur d'obéissance 0.5, pour priorité 2.0, exécutant le comportement $Comp1$ et de durée de violation 3. Si le contexte C est valide et que l'agent possède une intention portant sur le prédicat P, la norme est activable. Si l'agent utilise cette norme, il pourra être récompensée. Si par contre l'agent n'utilise pas cette norme, elle est violée pour 3 pas de calcul (sauf si elle est utilisée au cours de ces pas de calculs). Le fait que ces informations soient calculées par l'agent lui-même facilite un possible contrôle futur ; l'agent effectuant le contrôle n'a alors besoin que d'aller vérifier l'état des normes contrôlées, sans avoir à recalculer le contexte précédent pour avoir l'information sur la violation.

Ce processus est optionnel dans le sens où, si le modélisateur n'utilise pas de normes, leur statut ne peut pas être mis à jour. Le processus n'est donc pas utilisé par l'agent.

4.3 Discussions et conclusion

Cette dernière section a pour but de revenir sur l'ensemble de l'architecture et d'en discuter les détails qui n'ont pu être évoqués précédemment. Une première sous-section discute la façon dont les éléments proposés ont été élaborés, une seconde sous-section conclut le chapitre.

4.3.1 Discussion

Les différentes équations proposées pour le calcul des paramètres, des intensités et décroissances émotionnelles ou pour l'évolution des relations sociales ont été définies de sortes à ce qu'elles soient liées au plus faible nombre de paramètres du modèle OCEAN et de la façon la plus simple possible. En réduisant le nombre de paramètres que le modélisateur doit entrer, la phase

de paramétrisation d'un modèle utilisant BEN est simplifiée, ramenant le problème aux cinq dimensions du modèle OCEAN.

Nous avons fait le choix, pour la plupart des relations définies avec le modèle OCEAN, d'utiliser des fonctions linéaires pour des raisons de simplicité. Cependant, les relations linéaires ne paraissent pas satisfaisantes pour des paramètres comme l'obéissance où les gens ont tendance à être très obéissant, même avec une personnalité dans la moyenne [BBC+15]. Pour modéliser cela, l'équation (4.5) utilise une racine carrée.

Les équations pour le calcul initial de l'intensité des émotions créées par l'architecture, présentées en section 4.2.2, peuvent être simplifiées dans leur écriture ; elles sont présentées de sorte à souligner la façon dont elles ont été créées. Ces équations sont composées par un terme directement lié à l'état mental cognitif impliqué puis sont pondérées par une dimension issue de la personnalité ; cette dimension ayant sa valeur neutre à 0.5 comme l'explique la section 3.2.1. De plus, pour chaque émotion, une seule dimension de la personnalité est mise à contribution, celle étant le plus proche sémantiquement de l'émotion, toujours en ayant à l'esprit le principe de parcimonie.

Les équations pour l'évolution des dimensions des relations sociales sont développées pour respecter les notions psychologiques sous-jacentes mais aussi pour garder ces dimensions dans leurs limites (-1 et 1 pour l'appréciation et la dominance, 0 et 1 pour la solidarité et la familiarité). Les dimensions de la personnalité ont été intégrées, une fois encore, pour reproduire le fait que ces évolutions sont différentes pour chaque personne. La dimension de neurotisme est utilisée pour les parties d'équation liées aux émotions alors que la dimension d'ouverture est utilisée pour la partie en lien avec les similarités et les oppositions d'états mentaux cognitifs (quelqu'un d'ouvert d'esprit accordera moins d'importances à ce sujet comparé à quelqu'un de fermé d'esprit).

Chaque processus de l'architecture BEN possède un ordre d'exécution, comme le montre le diagramme d'activité en figure 4.1. Autrement dit, les processus de chaque module ne sont pas exécutés en même temps. Dans le module de perception, c'est le modélisateur qui décide de l'ordre des processus. Toutefois, un ordre logique serait de commencer par ajouter des croyances afin de mettre à jour les connaissances de l'agent puis d'effectuer la contagion émotionnelle, la création des relations sociales et le contrôle des normes, qui pourront s'effectuer dans un contexte à jour avec l'environnement.

Arrive ensuite la deuxième partie de l'architecture, s'occupant de modifier les connaissances de l'agent. Ici, les règles d'inférences sont exécutées en premier, permettant de modifier les bases d'états mentaux cognitifs, avant les autres processus. Ensuite sont exécutées les lois, puis le moteur émotionnel et enfin la mise à jour des relations sociales. L'étape de prise de décision, correspondant au troisième module de l'architecture, commence par vérifier si l'agent possède un système normatif. Les normes et obligations sont donc prioritaires sur la cognition. Une fois un plan ou une norme sélectionné, l'agent agit sur l'environnement. La dernière étape d'évolution temporelle dégrade d'abord les états mentaux cognitifs, puis les émotions et met enfin à jour le statut des normes.

L'avantage principal de BEN reste sa modularité. Dans ce chapitre, tous les processus sont expliqués en détail mais, à part le moteur cognitif, ils sont tous optionnels. Un modélisateur peut "débrancher" n'importe quel module optionnel de l'architecture sans avoir à se préoccuper du reste, l'architecture continuera de fonctionner correctement. Si un utilisateur estime que les normes et les obligations n'ont pas de sens vis à vis du problème modélisé, les processus liés aux normes et aux obligations peuvent être "débranchées" sans pour autant arrêter l'architecture de fonctionner. Un utilisateur expert peut aussi décider de changer certains modules de l'architecture. Par exemple, le moteur émotionnel peut être remplacé par un autre basé sur une autre théorie émotionnelle.

Cette modularité signifie que tous les composants cognitifs, affectifs et sociaux sont liés entre eux mais ne sont pas inter-dépendants. Les émotions ou les relations sociales peuvent être utilisées dans la définition du contexte des plans ou des normes, reliant toutes ces notions ensemble afin de créer un comportement plus riche. La seule partie minimale obligatoire est la définition

d'un désir et d'un plan permettant d'y répondre lorsque ce désir deviendra une intention.

4.3.2 Conclusion

Comme l'explique le chapitre 1, la simulation sociale a besoin d'outils permettant de simuler des acteurs humains. Pour résoudre ce problème, l'architecture BEN proposée dans ce chapitre intègre de nombreuses dimensions cognitives, affectives et sociales, en s'appuyant sur un formalisme, présenté dans le chapitre 3, permettant de faire interagir ces notions.

Ainsi, BEN propose des concepts de haut niveau qui viennent se greffer à une architecture BDI. Il s'agit donc d'une architecture cognitive qui est améliorée pour intégrer différents processus, venant modifier la prise de décision. Celle-ci n'est donc plus simplement rationnelle, elle dépend aussi de notions affectives comme les émotions, la contagion émotionnelle ou la personnalité mais aussi d'un contexte social, pris en compte à travers des relations sociales et un système normatif.

Pour résumer, BEN est une proposition d'architecture comportementale connectant plusieurs dimensions cognitives, affectives et sociales, les faisant interagir afin de créer le comportement d'un agent simulant un acteur humain. Néanmoins, cette architecture est indépendante de tout cas d'étude, ce qui signifie qu'elle peut s'adapter à différents contextes mais aussi évoluer pour accueillir de nouvelles dimensions et de nouveaux processus dans chacun de ces sous-modules.

Troisième partie

Mise en application de BEN

Chapitre 5

Implémentation de BEN dans GAMA

Sommaire

L'architecture BEN présentée dans le chapitre 4 a été implémentée dans la plateforme de modélisation et de simulation GAMA [GTG+13] pour faciliter son utilisation par la communauté des simulations sociales. Ce chapitre présente la façon dont BEN a été implémentée mais aussi les apports à la plateforme GAMA, en termes d'éléments de langages de programmation, réalisés pour faciliter l'utilisation d'une telle architecture.

Comme l'explique le chapitre 1, la réalisation de simulations sociales passe majoritairement par l'utilisation d'une plateforme dédiée, permettant à la fois de modéliser le problème étudié selon le paradigme agent mais aussi de simuler ce modèle et d'en extraire des résultats. Parmi ces plateformes, présentées en section 1.2.2, seule GAMA possède une architecture cognitive [TBC+16], fondée sur le modèle BDI, offrant aux modélisateurs la possibilité de créer des agents dont la prise de décisions repose sur une manipulation de concepts mentaux de haut niveau. Cette architecture BDI a été revue et améliorée lors de l'implémentation de BEN ce qui en fait maintenant une partie indissociable de l'architecture BEN dans GAMA.

GAMA est une plateforme de simulation à base d'agents avec son propre langage, le GAML, son compilateur et son environnement de programmation. Le but de GAMA est de faciliter l'accès aux simulations à base d'agents pour des personnes n'étant pas informaticiennes tout en proposant des outils de modélisation et d'analyse de pointe. C'est avec cette approche qu'est réalisée l'implémentation de BEN, en proposant des outils permettant de développer des comportements complexes, tout en les rendant le plus simple d'accès possible.

Ce chapitre commence par la présentation de GAMA et de son langage de programmation, le GAML, dans la modélisation de système à base d'agents. Ensuite, l'ensemble des ajouts imposés par l'architecture BEN au GAML pour définir des comportements agents sont détaillés. La

deuxième section du chapitre revient sur les différents types de données ajoutés à la plateforme en vue de l'implémentation de BEN, la troisième section traite de l'implémentation des processus de l'architecture et la dernière section discute des choix réalisés pour effectuer cette implémentation.

5.1 La plateforme GAMA

Parmi les plateformes de simulation à base d'agent présentées en section 1.2.2, nous avons choisi d'implémenter BEN dans GAMA [GTG+13]. Cette section offre une présentation rapide de la plateforme et de son langage de programmation, le GAML, spécialement dans la définition du comportement des agents.

5.1.1 Présentation générale de GAMA

GAMA (GIS Agent-based Modeling Architecture) est une plateforme de modélisation et de simulation à base d'agents, développée depuis 2007, ayant pour objectif de permettre la définition de simulations à base d'agents sur de grandes échelles, incluant de grandes quantités de données et pour autant utilisable par un public qui n'est pas informaticien. Pour cela, GAMA repose sur un environnement de développement intégré (IDE) permettant de rapidement passer de la modélisation à la simulation, et vice-versa. Cet atout majeur permet à un modélisateur de définir le modèle d'une part, de le simuler aussitôt et d'arrêter la simulation pour repasser à la modélisation si des modifications doivent être effectuées, sans avoir quitter la plateforme ou avoir eu besoin de coupler plusieurs plateformes différentes ensemble.

GAMA dispose de nombreuses options de visualisation des résultats de la simulation mais aussi de multiples outils permettant de faciliter la définition du modèle. Ainsi, il est possible de charger différents types de fichiers (CSV, shapefile, OSM file, ...), de sauvegarder les résultats sous la forme de fichiers CSV, de lancer des simulations par lots avec des paramètres d'initialisation différents, de visualiser des cartes ou des graphiques, et ainsi de suite.

Enfin, GAMA dispose de son propre langage de programmation, le GAML, avec son compilateur, son interpréteur et son éditeur permettant de définir le modèle à simuler. Cet éditeur intègre tous les outils des IDE récents, comme la coloration du texte, l'auto-complétion, l'auto-compilation ou la capacité de commenter automatiquement plusieurs lignes de code. De plus, ce langage est soutenu par une documentation en ligne ainsi que des exemples directement intégrés à la plateforme, permettant aux nouveaux utilisateurs de plus facilement appréhender les spécificités du GAML.

5.1.2 Développer un comportement agent dans GAMA

Le GAML est un langage orienté agent qui prend ses racines dans les langages orientés objet comme Java ou Smalltalk en étendant les concepts liés à la programmation orienté objet avec les concepts liés aux agents (autonomie, hétérogénéité). De cette façon, les modèles sont descriptifs et conservent leur expressivité tout en bénéficiant de notions comme l'héritage ou le typage fort.

En GAML, un agent est une instance d'une classe plus générale appelée *species* dans laquelle sont définies un ensemble d'attributs sous la formes de variables informatiques, et un ensemble de comportements, sous la formes de *statements*. Ces *statements* représentent des procédures informatiques permettant de modifier les attributs de l'agent lui-même, les attributs d'autres agents ou l'environnement partagé. Chaque *statement* se compose de plusieurs *facets* représentant des options possibles de paramétrisation, et possiblement d'un bloc de code à exécuter.

Le modélisateur dispose aussi d'une bibliothèque de fonctions et procédures, appelés respectivement en GAML des *opérateur* et des *actions*, qui peuvent être complétées par des *actions* définies par le modélisateur. Certaines de ces fonctions et procédures sont disponibles dans GAMA, d'autres font partie de *skills*, des collections de procédures et d'attributs pouvant être chargés lors de la définition des *species*.

Par exemple, le comportement d'un agent dans GAMA est défini à travers des *statements* **reflex** comme celui présenté dans l'exemple de code 5.1. Le mot "reflex" indique le type de *statement*, le *facet* "when" permet de mettre une condition à l'activation de ce réflexe (ici, il faut que la variable "energy" de l'agent soit inférieure à 0) et le bloc de code indique ce que l'agent va faire si ce *statement* est activé (dans cet exemple, l'agent va exécuter l'action "die", qui va le retirer de la simulation). Lorsque l'agent est actif dans le pas de temps de calcul, tous ses réflexes activables (avec une condition vraie pour le *facet* "when") seront exécutés dans l'ordre dans lequel ils sont définis.

```
reflex die when: energy <= 0 {
    do die;
}
```

Code 5.1 – Exemple d'un *statement* reflex définissant le fait que l'agent va mourir s'il n'a plus d'énergie.

GAMA permet également au modélisateur d'ajouter pour chaque espèce d'agents une architecture de contrôle. Cette architecture va fournir aux agents de cette espèce de nouvelles variables pré-définies, de nouveaux *statements*, de nouvelles actions et de nouveaux opérateurs. De plus, lors de l'exécution de la simulation, l'architecture va venir imposer un ordre d'activation entre les différents *statements*, permettant de contrôler la prise de décision de l'agent. L'architecture BEN, développée dans le chapitre 4, a été implémentée, et est utilisable dans GAMA, sous la forme de l'architecture de contrôle *simple_bdi*.

5.2 Types de données liés à BEN

Le chapitre 3 présente et formalise différentes notions qui servent de base à l'architecture BEN. Certaines de ces notions représentent des types de données manipulés par l'architecture pour prendre une décision cognitive, affective et sociale. Cette section présente l'implémentation de ces nouveaux types de données dans GAMA, permettant de représenter le monde sous la forme de prédicats puis de manipuler ces prédicats à travers des états mentaux cognitifs. Un type de donnée représentant des émotions et un type de donnée représentant les relations sociales sont aussi ajoutés à la plateforme.

En plus de la représentation formelle de ces types de données, les conditions d'égalité entre deux prédicats, deux états mentaux cognitifs, deux émotions ou deux relations sociales sont également explicitées.

5.2.1 Représentation du monde avec des prédicats

L'environnement dans lequel évolue l'agent est composé de variables de différents types ; des entiers ou des réels pour des quantités, des chaînes de caractère, des listes, des types de données spécialement créés, et ainsi de suite. Conformément au chapitre 3, la définition de prédicats permet d'unifier les informations du monde sous un même type de donnée, manipulable par l'architecture dans les processus de prise de décision. Dans *simple_bdi*, les prédicats sont représentés par le type **predicate** qui se compose des éléments suivants :

— Un nom, sous la forme d'une chaîne de caractères.

— Un ensemble de valeurs, sous la forme d'un dictionnaires de valeurs, associant une chaîne de caractères à une valeur de n'importe quel autre type.

— Un agent ayant causé le prédicat.

— Une valeur de vérité, représentée par un booléen, permettant de simplement représenter un fait P ou son exact opposé non P. Par défaut, cette valeur est à vrai.

Parmi ces valeurs, seul le nom est obligatoire, les autres champs sont optionnels. Cela signifie qu'un prédicat possède au minimum un nom, permettant de l'identifier clairement, puis peut accueillir un ensemble de valeurs, de la taille souhaitée par le modélisateur, un agent ayant causé l'information et enfin une valeur de vérité. Le code 5.2 montre un exemple de définition d'un prédicat dans GAMA. Cette variable de type **predicate** a pour nom "sac", elle porte sur un prédicat dont le nom est "sac" possédant une valeur dont le nom est "contenance" et associé à l'entier 5.

```
predicate sac <- new_predicate("sac",["contenance"::5]);
```

Code 5.2 – Exemple de définition d'un prédicat

5.2.2 Prise de décision sur des états mentaux cognitifs

Les états mentaux cognitifs, décrits dans la section 3.1.2, sont implémentés dans *simple_bdi* à travers le type de donnée **MentalState**, qui se compose des éléments suivants :

— Une modalité, sous la forme d'une chaîne de caractères, indiquant le type de l'état mental cognitif parmi Belief, Desire, Intention, Uncertainty, Obligation et Ideal.

— Un objet sur lequel porte cet état mental cognitif. Cela peut être un prédicat, un autre état mental cognitif ou une émotion.

— Une valeur réelle, dont le sens dépend de la modalité, conformément au formalisme détaillé en section 3.1.2.

— Une durée de vie, représentée par un entier.

— L'agent possédant cet état mental cognitif, ce qui permet de différencier deux états mentaux cognitif en fonction de l'agent qui les possède, notamment ans le cas d'un test d'égalité entre deux états mentaux portant eux-même sur des états mentaux.

Parmi ces champs, la modalité est obligatoire, tout comme l'objet sur lequel porte l'état mental cognitif. Pour le reste, le champs valeur est initialisé à 1.0 par défaut et il n'y a pas de durée de vie (ce qui représente une durée de vie infinie). Une fois créés, les états mentaux cognitifs sont stockés dans des bases de connaissances, une par modalité possible, par chaque agent.

Le test d'égalité entre deux états mentaux cognitif est simple. Cela signifie que deux états mentaux cognitifs sont égaux si leurs champs sont tous égaux, sauf pour la valeur réelle et la durée de vie, qui ne sont pas inclus dans le test d'égalité.

Pour faciliter la manipulation des états mentaux cognitifs, des actions "add", "remove" et "get", détaillées dans le tableau 5.1, ont été développées. Ainsi, pour manipuler un état mental cognitif, l'utilisateur doit simplement renseigner l'objet sur lequel porte cet état mental cognitif et les champs optionnels souhaités (force, durée de vie) et *simple_bdi* ajoute, enlève ou de récupère l'état mental cognitif avec la modalité correspondante dans les bases de l'agent. Le code 5.3 montre un exemple d'utilisation de l'action $add\_desire()$ permettant d'ajouter un désir portant sur la variable de type predicate nommée "wander", définie au préalable.

```
do add_desire(wander);
```

Code 5.3 – Exemple d'ajout manuel d'un désir dans GAMA

5.2.3 Définition des émotions

L'implémentation des émotions suit le formalisme présenté en section 3.2.2. Les émotions sont implémentées dans GAMA à travers le type **emotion** qui se compose des éléments suivants :

— Un nom, sous la forme d'une chaîne de caractères, servant d'identifiant à l'émotion

TABLEAU 5.1 – Actions permettant de manipuler les états mentaux cognitifs de l'agent

nom	arguments	description
add_belief(), add_desire(), ...	prédicat, strength, lifetime	Permet d'ajouter un état mental cognitif (une croyance, une incertitude, un désir, une obligation ou un idéal) à propos d'un *prédicat* donné, avec pour valeur de force *strength* et pour durée de vie *lifetime*.
remove_belief(), remove_desire(), ...	prédicat	Permet de retirer un état mental cognitif (une croyance, une incertitude, un désir, une obligation, une intention ou un idéal) à propos d'un *prédicat* donné.
get_belief(), get_desire(), ...	prédicat	Permet de récupérer une instance d'un état mental cognitif (une croyance, une incertitude, un désir, une obligation, une intention ou un idéal) à propos d'un *prédicat* passé en argument.

— Un objet, de type predicate, sur lequel porte l'émotion.

— Une intensité réelle, supérieure à 0.

— Une valeur de décroissance réelle.

— Un agent causant l'émotion.

Seul le nom est obligatoire dans la définition d'une émotion dans GAMA ; les autres composantes sont optionnelles. Le test d'égalité entre deux émotions suit le principe du test d'égalité entre deux états mentaux cognitifs. Cela signifie que deux émotions sont égales si elles ont le même nom, et portent sur le même objet, quelque soit leur valeur d'intensité, de décroissance ou leur agent cause.

Comme pour les états mentaux cognitifs, chaque agent stocke ses émotions dans une base des émotions. Pour faciliter la gestion de cette base, des actions $add\_emotion()$, $remove\_emotion()$ et $get\_emotion()$ ont été développés, permettant d'ajouter, d'enlever ou de récupérer une émotion dans la base d'un agent. Le code 5.4 montre la définition d'une émotion de joie portant sur une objet de type predicate nommé "wander" puis l'ajout de cette émotion à la base des émotions de l'agent en cours d'exécution.

TABLEAU 5.2 – Actions permettant de manipuler les émotions de l'agent

nom	arguments	description
add_emotion()	émotion	Permet d'ajouter une *émotion*.
remove_emotion()	émotion	Permet de retirer une *émotion*.
get_emotion()	émotion	Permet de récupérer une instance correspondant à l'*émotion* passée comme argument.

```
emotion joie <- new_emotion("joie",wander);
do add_emotion(joie);
```

Code 5.4 – Exemple d'ajout d'une émotion de joie

5.2.4 Représentation des relations sociales

L'implémentation des relations sociales se fait en suivant le formalisme présenté dans la section 3.3.2 à travers le type **SocialLink** qui se compose des éléments suivantes :

— L'agent envers qui est définie le lien.

— Une valeur réelle d'appréciation.

— Une valeur réelle de dominance.

— Une valeur réelle de solidarité.

— Une valeur réelle de familiarité.

— Une valeur réelle de confiance.

Dans la définition de ces liens, seul l'agent envers qui est tourné le lien est obligatoire, les autres valeurs sont optionnelles. Chaque agent stocke les liens sociaux qu'il a avec les autres agents dans une base des liens sociaux, en ne conservant qu'un seul lien pour chaque agent. Le code 5.5 montre la façon dont une relation sociale peut être ajoutée dans *simple_bdi* avec un autre agent, nommé "friend" et défini préalablement. Les actions permettant de manipuler des relations sociales sont détaillées dans le tableau 5.3.

TABLEAU 5.3 – Actions permettant de manipuler les relations sociales de l'agent

nom	arguments	description
add_social_link()	lien social	Permet d'ajouter un *lien social.*
remove_social_link()	lien social	Permet de retirer un *lien social.*
get_social_link()	lien social	Permet de récupérer une instance du *lien social* passé en argument.

```
do add_social_link(new_social_link(friend));
```

Code 5.5 – Exemple d'ajout d'une relation sociale

5.3 Implémentation des processus de BEN

Les processus de l'architecture BEN pouvant être modifiés manuellement sont accessibles à l'utilisateur via la définition de *statements*, alors que les processus automatiques sont exécutés directement par la plateforme, si le modélisateur a indiqué qu'il les utilisait. L'implémentation de tous ces processus est décrite en détail dans cette section, en reprenant le découpage en modules utilisé dans la section 4.2.

5.3.1 Module de perception

La première étape de l'architecture BEN, comme expliqué dans la section 4.2.1, consiste à définir la perception de l'agent, permettant de mettre à jour les connaissances en fonction de l'environnement perçu. Pour définir cette perception, le modélisateur doit utiliser le *statement* **perceive** qui possède plusieurs *facets* expliqués dans le tableau 5.4.

Ce *statement* fonctionne de la façon suivante : pour chaque agent faisant partie de l'ensemble indiqué dans le *facet* **target** et se trouvant dans la géométrie décrite par le *facet* **in**, si la condition indiquée dans le *facet* **when** est vraie, que l'agent possède l'émotion renseignée dans le *facet* **emotion** avec une intensité supérieur à celle indiquée par le *facet* **threshold**, alors le bloc de code associé au *statement* est exécuté. La particularité du **perceive** est que le bloc de code est exécuté dans le contexte de l'agent perçu, ce qui signifie que le modélisateur a accès directement aux variables de l'agent perçu.

TABLEAU 5.4 – Présentation des *facets* du *statement* **perceive**

nom	type de donnée	obligatoire	description
target	liste d'agents	oui	L'ensemble des agents visés par la perception.
in	un réel ou une géométrie	non	Une géométrie à l'intérieur de laquelle sera effectuée la perception. L'utilisateur peut y renseigner un nombre positif indiquant un rayon de perception autour de l'agent percevant ou directement une géométrie.
when	booléen	non	Une condition permettant de n'effectuer la perception que selon certaines conditions.
emotion	émotion	non	Une émotion que l'agent doit ressentir pour effectuer la perception.
threshold	réel	non	Un seuil sur l'intensité émotionnelle du *facet* emotion. Si l'intensité est supérieure au seuil, la perception est exécutée.

Le code 5.6 montre un exemple de définition d'un *statement* perceive dans GAMA. Dans cet exemple, la perception se fait sur tous les agents appartenant à l'espèce *gold* dans un rayon représenté par la variable *viewdist*. Pour chaque agent de l'espèce *gold* se trouvant à une distance inférieure à *viewdist* par rapport à l'agent effectuant cette perception, le bloc de code associé est exécuté. Dans ce bloc de code, le modélisateur aura un accès direct aux variables associées aux agents de l'espèce *gold*.

```
perceive target:gold in:viewdist{
...
}
```

Code 5.6 – Exemple de définition d'une perception

Les *statements* décrits dans la suite de cette section doivent être nécessairement définis dans le bloc de code lié à une perception et seront exécutés dans le contexte des agents perçus (les variables appelées seront évaluées du point de vue de l'agent perçu, sauf mention contraire dans la définition des *facets* de chaque *statement*).

Ajout de croyances et d'incertitudes

Le rôle premier de la phase de perception est de mettre à jour les connaissances de l'agent réalisant la perception; cela commence par l'ajout de croyances et d'incertitudes par rapport à l'environnement perçu. L'opération consiste alors à traduire les variables de l'environnement en prédicats, puis à intégrer ces prédicats dans des états mentaux cognitifs ajoutés à l'agent réalisant la perception. Pour faciliter cette opération, nous avons défini un *statement* appelé **focus** dont les différentes *facets* sont expliquées dans le tableau 5.5.

En parallèle de ce *statement,* il est aussi possible de gérer manuellement les états mentaux cognitifs de l'agent effectuant la perception. Pour cela, il est nécessaire de revenir dans le contexte de l'agent percevant (et non de celui perçu) à l'aide du *statement ask* de la plateforme GAMA qui permet de demander à un agent d'exécuter un ensemble d'instructions.

TABLEAU 5.5 – Présentation des *facets* du *statement* **focus**

nom	type de donnée	obligatoire	description
id	chaîne de caractères	oui	Représente le nom qui sera donné au prédicat créé dans l'opération.
var	une variable de l'agent perçu	non	Une variable de l'agent perçu dont la valeur sera stockée dans l'ensemble de valeurs du prédicat créé, associée au nom "value_var".
expression	une expression mathématique	non	Une expression mathématique permettant d'altérer la valeur de la variable stockée, permettant de simuler une imprécision de la perception.
truth	booléen	non	Valeur de vérité de l'état mental cognitif créé. Par défaut, l'état mental créé est vrai.
lifetime	entier	non	Durée de vie de l'état mental cognitif créé.
agent_cause	agent	non	Un agent considéré comme responsable de l'information ajoutée. Par défaut, il s'agit de l'agent perçu.
strength	réel	non	Valeur de force associée à l'état mental cognitif créé.
is_uncertain	booléen	non	Indique si l'état mental cognitif créé est une croyance (à faux) ou une incertitude (à vrai). Par défaut à faux.
belief, desire, uncertainty, ideal	état mental cognitif	non	Permettent d'ajouter des croyances ou des incertitudes sur les états mentaux cognitifs de l'agent perçu, en renseignant la connaissance à observer.
emotion	émotion	non	Permet d'ajouter des croyances ou des incertitudes sur les émotions de l'agent perçu, en renseignant l'émotion à observer.
when	booléen	non	Une condition permettant de n'effectuer ce processus que selon certaines conditions.

Le code 5.7 montre un exemple de perception utilisant à la fois le *statement* focus pour ajouter une croyance mais aussi une manipulation manuelle retirant une intention. Ce code signifie que, si un agent de l'espèce *gold* se trouve à une distance inférieure à *viewdist* d'un agent A effectuant la perception, une croyance portant sur un prédicat dont le nom est "location_gold" est ajouté à l'agent A. Le prédicat de la croyance ajoutée contiendra l'emplacement de l'agent *gold* perçu (la variable *location* fait bien référence à l'emplacement de l'agent *gold*) associé au nom "location_value". Ensuite, l'agent A va se retirer une intention portant sur une variable de type **predicate** nommée "wander" tout en conservant le désir lié à ce prédicat (grâce à l'argument *false* passé dans la procédure *remove_intention*).

```
perceive target:gold in:viewdist {
    focus id:"location_gold" var:location;
    ask myself {do remove_intention(wander, false);}
}
```

Code 5.7 – Exemple de modification des bases de connaissance lors d'une perception

Contagion émotionnelle

La définition d'une contagion émotionnelle pendant la perception d'un autre agent possédant des émotions se fait à travers le *statement* **emotional_contagion**, composé des *facets* décrits dans le tableau 5.6.

TABLEAU 5.6 – Présentation des *facets* du *statement* **emotional_contagion**

nom	type de donnée	obligatoire	description
emotion_-detected	émotion	oui	L'émotion devant être détectée chez l'agent perçu pour que la contagion ait lieu.
emotion_-created	émotion	non	L'émotion créée lors de la contagion émotionnelle, dans le cas où la contagion crée une émotion différente de l'émotion détectée.
charisma	réel	non	Une valeur de charisme pour la contagion émotionnelle. Par défaut, il s'agit de la valeur de charisme de l'agent perçu, présentée et calculée en section 4.1.2.
receptivity	réel	non	Une valeur de réceptivité pour la contagion émotionnelle. Par défaut, il s'agit de la valeur de réceptivité émotionnelle de l'agent effectuant la perception, comme indiqué en section 4.1.2.
threshold	réel	non	Une valeur utilisée comme seuil pour effectuer la contagion seulement si les valeurs de charisme et de réceptivité émotionnelle sont suffisantes, conformément au formalisme présenté en section 3.2.2. Par défaut, cette valeur est à 0.25.
intensity	agent	non	Une valeur permettant d'indiquer l'intensité de l'émotion créée chez l'agent effectuant la perception.
decay	réel	non	Une valeur permettant d'indiquer la valeur de décroissance de l'émotion créée chez l'agent effectuant la perception.
when	booléen	non	Une condition permettant de n'effectuer cette contagion que selon certaines conditions.

Si le *facet* **emotion_created** n'est pas utilisé, la contagion émotionnelle consistera dans la copie de l'émotion détectée, si celle-ci est détectée et si les valeurs de charisme et de réceptivité émotionnelle sont suffisantes, comme expliqué dans le formalisme 3.2.2. S'ils ne sont pas utilisés, les *facets* **charisma**, **receptivity** et **threshold** prennent leurs valeurs par défaut. Si les *facets* **intensity** et **decay** ne sont pas utilisés, le calcul de l'intensité et de la valeur de décroissance de l'émotion créée suivent les règles exprimées en section 4.2.1.

Le code 5.8 montre un exemple de contagion émotionnelle au cours d'une perception. Si des agents appartenant à l'espèce *people* se trouvent à une distance inférieure à *other_distance* de l'agent effectuant la perception, alors deux contagions émotionnelles ont lieu. La première contagion émotionnelle porte simplement sur la détection d'une émotion nommée "joy". Si cette émotion est détectée dans l'agent perçu, elle est ajoutée à l'agent effectuant la perception, si la multiplication des valeurs de charisme et de réceptivité est au dessus de 0.25. La seconde contagion émotionnelle indique qu'une émotion nommée "fear" est créée lorsque l'émotion nommée "fearConfirmed" est détectée chez l'agent perçu, toujours en fonction des valeurs de charisme et de réceptivité des agents.

```
perceive target:people in:other_distance {
    emotional_contagion emotion_detected: joy;
    emotional_contagion emotion_detected:ferConfirmed emotion_created:fear;
}
```

Code 5.8 – Exemple de contagions émotionnelles lors d'une perception

Création de relations sociales

La création de relations sociales dynamiques se fait dans une perception à l'aide du *statement* **socialize** qui comporte les *facets* détaillés dans le tableau 5.7

TABLEAU 5.7 – Présentation des *facets* du *statement* **socialize**

nom	type de donnée	obligatoire	description
liking	réel	non	Une valeur entre -1 et 1 indiquant la valeur d'appréciation initiale de la relation sociale créée.
dominance	réel	non	Une valeur entre -1 et 1 indiquant la valeur de dominance initiale de la relation sociale créée.
solidarity	réel	non	Une valeur entre 0 et 1 indiquant la valeur de solidarité initiale de la relation sociale créée.
familiarity	réel	non	Une valeur entre 0 et 1 indiquant la valeur de familiarité initiale de la relation sociale créée.
trust	réel	non	Une valeur entre -1 et 1 indiquant la valeur de confiance initiale de la relation sociale créée.
agent	agent	non	L'agent envers qui est créée la relation sociale. Par défaut, il s'agit de l'agent perçu.
when	booléen	non	Une condition permettant de n'effectuer la socialisation que selon certaines conditions.

Par défaut, c'est à dire en n'utilisant aucun des *facets*, le *statement* **socialize** crée, chez l'agent réalisant la perception, une relation sociale avec l'agent perçu dont chaque champ est initialisé à une valeur de 0. S'il existe déjà une relation sociale avec l'agent perçu, le *statement* **socialize** n'agira plus comme un processus d'initialisation de relation sociale.

Le code 5.9 représente un exemple d'utilisation du *statement* **socialize**. Si des agents appartenant à l'espèce *people* se trouvent à une distance inférieure à *other_distance* de l'agent effectuant la perception, alors une relation sociale neutre est créée avec chacun d'entre eux.

```
perceive target:people in:other_distance {
    socialize;
}
```

Code 5.9 – Exemple d'une création dynamique de relations sociales lors d'une perception

TABLEAU 5.8 – Présentation des *facets* du *statement* **enforcement**

nom	type de donnée	obligatoire	description
norm	chaîne de caractères	non	Le nom de la norme à contrôler chez l'agent perçu.
obligation	prédicat	non	Un prédicat sur lequel porte une obligation à contrôler.
law	chaîne de caractères	non	Le nom de la loi à contrôler chez l'agent perçu.
sanction	chaîne de caractères	non	Le nom de la sanction à appliquer par l'agent effectuant la perception, en cas de sanction lors du contrôle.
reward	chaîne de caractères	non	Le nom de la sanction à appliquer par l'agent effectuant la perception, en cas de récompense lors du contrôle.
when	booléen	non	Une condition permettant de n'effectuer ce contrôle que selon certaines conditions.

Contrôle des normes

Le contrôle des normes, lois et obligations, comme expliqué en section 4.2.1, se fait dans *simple_bdi* lors d'une perception, selon le *statement* **enforcement** qui possède les *facets* détaillés dans le tableau 5.8.

Si tous les *facets* de ce *statement* sont optionnels, ils ne le sont pas tous en même temps. Ainsi, pour que le contrôle ait un sens, il faut qu'il y ait une modalité à contrôler (parmi les *facets* **norm**, **obligation** et **law**) mais aussi au moins une sanction ou une récompense. Le fonctionnement de ce contrôle est expliqué en détail dans la section 4.2.1.

La définition de contrôles nécessite de définir des sanctions, qui serviront à sanctionner ou à récompenser l'agent contrôlé. Cela se fait par le biais du *statement* **sanction** qui n'est composé que d'une chaîne de caractères en guise de nom puis d'un bloc de code exécuté lors du contrôle du système normatif.

Le code 5.10 montre un exemple de définition de contrôles au cours d'une perception, ainsi qu'un exemple de définition d'une sanction. Les contrôles sont effectués si des agents de l'espèce *miner* se trouvent à une distance *viewdist* de l'agent effectuant la perception. Chaque agent perçu est stocké dans la variable *agent_perceived* pour pouvoir être utilisé dans les sanctions et les récompenses. Le premier contrôle porte sur une loi nommée "working" et ne fait que sanctionner l'agent perçu si la loi est violée. Le deuxième contrôle se concentre sur l'obligation portant sur le prédicat *has_gold* et possède une sanction et une récompense distinctes. Le troisième contrôle s'intéresse à une norme nommée "share_information" et ne fait que récompensée l'agent perçu si la norme est suivie. Cette récompense, qui est définie par la suite hors de la perception, modifie la valeur de confiance avec l'agent perçu.

```
perceive target:miner in:viewdist {
    agent_perceived<-self;
    enforcement law:"working" sanction:"sanctionToLaw";
    enforcement obligation:has_gold
    sanction:"sanctionToObligation"
    reward:"rewardToObligation";
    enforcement norm:"share_information" reward:"rewardToNorm";
}

sanction rewardToNorm{
    do change_liking(agent_perceived,0.1);
}
```

Code 5.10 – Exemple de définition de contrôles du système normatif

5.3.2 Mise à jour des connaissances de l'agent

Le deuxième module principal de BEN porte sur la modification des connaissances de l'agent une fois la perception effectuée. Le module comprend en particulier l'activation de règles d'inférences et de lois qui devront être définies par le modélisateur à l'aide de *statements* dédiés, détaillés dans les sous-sections ci-dessous. Les deux autres processus du module, la création des émotions et la mise jour des relations sociales, sont automatiques, ce qui signifie que l'utilisateur indique simplement s'il souhaite les utiliser ou non, sans avoir à les définir ou à les paramétrer.

La création automatique des émotions se fait en suivant les principes exposés en section 4.2.2 : les vingt règles proposées sont testées afin de créer des émotions par rapport aux nouvelles connaissances acquises lors de la perception. Dans la pratique, cette création automatique des émotions se fait lors de l'ajout d'une croyance ou d'une incertitude. Cela signifie que les émotions ne sont créées qu'au moment où une connaissance est ajoutée dans les bases de l'agent afin de ne pas créer en boucle une émotion, simplement parce que des faits déclenchant une émotion sont toujours connus de l'agent, sans avoir évolués.

Si l'émotion créée par le moteur émotionnel était déjà présente dans les bases de connaissance de l'agent, les deux émotions fusionnent en une seule émotion. L'intensité de cette émotion fusionnée correspond à la somme des intensités des deux émotions en jeu, alors que sa valeur de décroissance correspond à la valeur de décroissance de l'émotion en jeu possédant l'intensité maximale.

L'utilisateur peut choisir de ne pas utiliser le moteur émotionnel permettant de créer automatiquement les vingts émotions issues du modèle OCC. Par défaut, dans *simple_bdi*, le moteur émotionnel n'est pas activé afin de ne pas alourdir les calculs de prise de décision si le modélisateur ne souhaite pas utiliser les émotions dans la modélisation des agents. Afin d'activer ce moteur, l'utilisateur doit indiquer que la variable booléenne **use_emotions_architecture**, associée à chaque agent utilisant l'architecture BEN, prend la valeur vrai.

Le moteur social permettant de mettre à jour les relations sociales d'un agent en fonction de ses connaissances acquises par la perception fonctionne comme expliqué en section 4.2.2. Du point de vue de l'implémentation, les quatre règles d'évolution des dimensions d'une relation sociale sont activées à chaque pas de temps par l'intermédiaire du *statement* **socialize**, présenté en section 5.3.1. De cette façon, les relations sociales entre les agents sont mises à jour lorsque ceux-ci se perçoivent, et peuvent alors prendre conscience des actions de l'un sur l'autre.

En pratique, la première fois que deux agents se perçoivent et utilisent le *statement* **socialize**, une relation sociale va se créer. Lors des perceptions suivantes, le **socialize** va vérifier s'il existe déjà une relation sociale avec l'agent perçu. Si c'est le cas, les formules d'évolution des dimensions d'une relation sociale, détaillées en section 4.2.2 sont exécutées.

Comme pour le moteur émotionnel, le moteur social est optionnel dans l'architecture BEN.

Par défaut, le moteur social n'est pas activé dans l'implémentation de l'architecture BEN dans GAMA. Pour l'activer, en plus d'utiliser le *statement* **socialize**, l'utilisateur doit indiquer que la variable booléenne **use_social_architecture**, associée à chaque agent utilisant l'architecture BEN, prend la valeur vrai.

Le reste de la section détaille les ajouts au langage de programmation nécessaire pour définir des règles d'inférence et des lois, représentant les deux autres processus du module de mise à jour des connaissances de l'agent dans l'architecture BEN.

Définir des règles

La définition de règles d'inférences, permettant de modifier les connaissances de l'agent en fonction des ajouts réalisés pendant la perception, se fait dans GAMA par le biais du *statement* **rule**, dont les *facets* sont détaillés dans le tableau 5.9.

Les *facets* des règles ne sont pas tous optionnels en même temps. Un règle d'inférence peut être vu comme une règle "si-alors" : si l'agent possède la connaissance indiquée, alors, il crée ou retire une autre connaissance. Les nombreux *facets* permettent de définir différents types de règles, plus ou moins complexes, de sorte à s'ajuster au plus de cas d'utilisation possibles.

Le code 5.11 montre un exemple de règle d'inférence. Si l'agent possède une croyance portant sur le prédicat identifié par la variable *has_gold*, alors il s'ajoute un désir portant sur le prédicat identifié par la variable *sell_gold*, en lui associant une valeur de priorité de 3.0.

```
rule belief: has_gold new_desire: sell_gold strength: 3.0;
```

Code 5.11 – Exemple de définition d'une règle d'inférence

Définir des lois

La définition de lois, conformément à ce qui est expliqué en section 4.2.2, se fait dans GAMA par le biais du *statement* **law** qui possède les *facets* détaillés dans le tableau 5.10.

Comme pour les règles d'inférence, les *facets* d'une loi ne sont pas tous optionnels en même temps. Il est nécessaire d'avoir une condition de déclenchement ainsi qu'une obligation à créer. Aussi, le déclenchement d'une loi retire toutes les intentions de l'agent, de sorte que l'agent puisse considérer, aussitôt dans sa prise de décision, l'obligation ajoutée par la loi.

Le code 5.12 montre un exemple de définition d'une loi dans GAMA. Si l'agent possède une croyance sur le prédicat identifié par la variable *mine_location* et que sa valeur d'obéissance est supérieure à la valeur identifiée par la variable *thresholdLaw*, alors une obligation lui est ajoutée, portant sur le prédicat *has_gold* avec une priorité de valeur 2.0.

```
law working belief: mine_location new_obligation: has_gold
strength: 2.0 threshold:thresholdLaw;
```

Code 5.12 – Exemple de définition d'une loi dans GAMA

5.3.3 Prise de décision cognitives et normatives

La dernière partie paramétrable par l'utilisateur de l'architecture BEN concerne la prise de décision à travers le moteur cognitif et le moteur normatif, comme expliqué en section 4.2.3. Si ces deux moteurs sont exécutés automatiquement, ils s'appuient sur les concepts de plans d'action et de normes qui doivent être définis par le modélisateur, en accord avec leur formalisme respectif, présenté en section 3.1.2 pour les plans et en section 3.3.1 pour les normes.

TABLEAU 5.9 – Présentation des *facets* du *statement* **rule**

nom	type de donnée	obligatoire	description
belief, desire, uncertainty, ideal, obligation	prédicat	non	Un prédicat sur lequel l'agent possède une connaissance, permettant de déclencher la règle.
emotion	émotion	non	Une émotion qui déclenche la règle si elle est possédée par l'agent.
beliefs, desires, uncertainties, ideals, obligations	liste de prédicats	non	L'ensemble des connaissances que l'agent doit posséder pour déclencher la règle.
emotions	liste d'émotions	non	Un ensemble d'émotions nécessaires pour déclencher la règle.
new_belief, new_desire, new_uncertainty, new_ideal	prédicat	non	Permet de créer une connaissance sur un prédicat.
new_emotion	émotion	non	L'émotion à créer par la règle.
new_beliefs, new_desires, new_uncertainties, new_ideals	liste de prédicats	non	Permet de créer des connaissances sur un ensemble de prédicats, une connaissance par prédicat.
new_emotions	liste d'émotions	non	Un ensemble d'émotions créées par la règle.
remove_belief, remove_desire, remove_uncertainty, remove_ideal, remove_obligation, remove_intention	prédicat	non	Permet de retirer une connaissance sur un prédicat particulier.
remove_emotion	émotion	non	Une émotion à retirer.
remove_beliefs, remove_desires, remove_uncertainties, remove_ideals, remove_obligations, remove_intentions	liste de prédicats	non	Permet de retirer plusieurs croyances portant sur des prédicats différents.
remove_emotions	liste d'émotions	non	Permet de retirer plusieurs émotions.
threshold	réel	non	Un seuil indiquant le niveau d'intensité minimal d'une émotion utilisée comme déclencheur de la règle d'inférence.
strength	réel	non	Valeur associée à l'état mental cognitif ajouté.
lifetime	entier	non	durée de vie de l'état mental cognitif créé.
all	booléen	non	Permet de créer un état mental cognitif par état mental détecté, en copiant les valeurs des prédicats (par exemple, pour créer un désir par croyance portant sur un incendie en stockant sa position, en sachant qu'il y a plusieurs incendies donc plusieurs croyances).
when	booléen	non	Une condition permettant de n'exécuter la règle que selon certaines conditions.

TABLEAU 5.10 – Présentation des *facets* du *statement* **law**

nom	type de donnée	obligatoire	description
belief	prédicat	non	Un prédicat sur lequel l'agent possède une croyance, permettant de déclencher la loi.
beliefs	liste de prédicats	non	Un ensemble de prédicats sur lesquels l'agent possède des croyances, nécessaires pour déclencher la loi.
new_obligation	prédicat	non	Permet de créer une obligation sur un prédicat particulier.
new_obligations	liste de prédicats	non	Permet de créer un ensemble d'obligations portant sur un ensemble de prédicats.
threshold	prédicat	non	Seuil entre 0 et 1. Si la valeur d'obéissance de l'agent est supérieure à ce seuil, la loi est exécutée.
strength	émotion	non	La priorité associée à l'obligation créée par la loi.
lifetime	liste de prédicats	non	Durée de vie de l'obligation créée par la loi.
all	booléen	non	Permet de créer un ensemble d'obligations correspondant à un ensemble de croyances portant sur la même information mais avec des valeurs différentes.
when	booléen	non	Une condition permettant de n'exécuter la loi que selon certaines conditions.

Les moteurs cognitifs et normatifs fonctionnent comme expliqué en section 4.2.3 et permettent à l'agent de prendre une décision sur l'action à effectuer en fonction de l'état de ses connaissances. Cette prise de décision est, par défaut, déterministe. Cela signifie que le désir de plus haute priorité est sélectionné en tant qu'intention et que le plan, ou la norme, de plus haute priorité est sélectionné comme action à exécuter. Pour autant, il est possible de rendre cette prise de décision probabiliste (les priorités des désirs et des plans/normes sont alors considérés comme des probabilités) en passant la variable booléenne de l'agent **probabilistic_choice** à vrai.

Comme expliqué en section 4.2.3, les agents qui utilisent l'architecture BEN peuvent remettre en question leur intention courante ou leur plan en cours, de façon aléatoire. Par défaut, cette option est reliée à la personnalité de l'agent, comme détaillé en section 4.1.2. Pour autant, si un modélisateur souhaite définir ses propres probabilités de conserver les intentions et les plans, indépendamment de la personnalité, il peut le faire. Pour cela, le modélisateur doit de passer la variable **use_persistence** à vrai, puis indiquer les valeurs souhaitées pour les variables **persistence_coefficient_intentions** et **persistence_coefficient_plans**. Chaque coefficient représente une probabilité, entre 0 et 1, que l'agent remette en cause son intention courante ou son plan/norme en cours à chaque pas de temps, indépendamment de l'évolution de ses connaissances.

Le reste de cette section revient en détail sur la définition des plans et des normes, avec les *statements* associés.

Plans

Les plans représentent une suite d'instructions à exécuter en réponse à l'intention courante. Dans *simple_bdi*, la définition de plans se fait via le *statement* **plan** dont les *facets* sont décrits dans le tableau 5.11.

Lors de la prise de décision, seuls les plans répondant à l'intention en cours sont considérés. Le moteur cognitif évalue ensuite les *facets* **when**, **emotion** et **threshold**, afin de ne conserver que les

TABLEAU 5.11 – Présentation des *facets* du *statement* **plan**

nom	type de donnée	obligatoire	description
name	chaîne de caractères	oui	Le nom du plan d'action.
intention	prédicat	oui	Permet de répondre à une intention courante portant sur un prédicat précis.
finished_when	booléen	non	Indique le moment où le plan est considéré comme terminé.
priority	réel	non	Une valeur permettant de classer des plans répondant à la même intention.
emotion	émotion	non	Le plan est exécuté si l'agent possède l'émotion renseignée.
threshold	réel	non	Un seuil sur le *facet* emotion. Le plan est exécuté si l'intensité de l'émotion est supérieur à ce seuil.
instantaneous	booléen	non	Indique si le plan doit être calculé de façon instantanée, permettant d'exécuter plusieurs plans ou normes dans un même pas de temps.
when	booléen	non	Une condition permettant de n'exécuter le plan que dans certains cas.

plans applicables dans le contexte actuel, en fonction des connaissances de l'agent. Si l'ensemble des plans exécutables est vide, l'intention courante est remise en cause, de sorte à choisir une intention qui possède au moins un plan exécutable. S'il existe plusieurs plans exécutables pour l'intention courante dans le contexte courant, la valeur de priorité permet de classer ces plans, soit en sélectionnant celui de plus haute priorité, soit en utilisant les priorités comme probabilités, si la variable **probabilistic_choice** est à vraie. Une fois le plan sélectionné, le bloc de code associé est exécuté.

Le code 5.13 montre un exemple de définition d'un plan permettant à l'agent de se déplacer de façon aléatoire. Le plan porte le nom *letsWander* et répond à une intention portant sur le prédicat identifié par la variable *find_gold*. Une fois sélectionné, ce plan consiste à utiliser l'action *wander*, déjà présente dans GAMA, indiquant à l'agent qu'il se déplace d'une distance unitaire dans une direction aléatoire.

```
plan letsWander intention:find_gold {
    do wander;
}
```

Code 5.13 – Exemple de définition d'un plan dans GAMA

Normes

Comme expliqué en section 3.3.1, le concept de norme représente ici une suite d'instructions à effectuer pour répondre à une intention, mais pouvant ne pas être choisie par l'agent par désobéissance. Ces normes sont définies dans *simple_bdi* à travers le *statement* **norm** composé des *facets* détaillés dans le tableau 5.12.

Les normes sont mises en compétition avec les plans afin de répondre aux intentions courantes. Pour autant, les normes différencient les intentions issues des désirs des intentions issues

TABLEAU 5.12 – Présentation des *facets* du *statement* **norm**

nom	type de donnée	obligatoire	description
name	chaîne de caractères	oui	Le nom de la norme.
intention	prédicat	non	Permet de répondre à une intention courante portant sur un prédicat précis.
obligation	prédicat	non	Permet de répondre à une intention courante issue d'une obligation et portant sur un prédicat précis.
finished_when	booléen	non	Indique le moment où la norme est considérée comme terminée.
priority	réel	non	Une valeur permettant de classer des normes répondant à la même intention.
threshold	réel	non	Un seuil d'obéissance. La norme est exécutée si la valeur d'obéissance de l'agent est supérieure à ce seuil.
instantaneous	booléen	non	Indique si la norme doit être calculée de façon instantanée, permettant d'exécuter plusieurs plans ou normes dans un même pas de temps.
when	booléen	non	Une condition permettant de n'exécuter la norme que dans certains cas.

des obligations. Cette distinction permet de créer différents types de contrôles, et donc des systèmes normatifs plus ou moins complexes, comme expliqué en section 4.2.1.

Contrairement aux plans, il est possible de désobéir aux normes. Cela signifie qu'en plus des conditions d'activation liées aux intentions, aux obligations et au contexte de l'agent, il est possible de ne pas suivre une norme dans le cas d'une valeur d'obéissance inférieure au seuil de la norme. Dans ce cas, l'agent peut sélectionner un plan répondant à son intention, activable dans les mêmes conditions, puis considérer sa norme comme violée.

Le code 5.14 montre un exemple de définition d'une norme répondant à une intention portant sur le prédicat *share_information*, avec un seuil d'obéissance identifié par la variable *thresholdNorm*. Cette norme d'exemple est instantanée ce qui signifie que si elle est choisie, elle est exécutée puis l'agent effectue de nouveau son cycle de prise de décision, sans avoir passé un pas de temps. Cette notion de plan, ou de norme, instantané permet de réaliser des actions de durée négligeable par rapport au pas de temps considéré. Par exemple, dans le modèle de Truong [TTG+15] qui s'intéresse aux changements de production des agriculteurs dans le Delta du Mékong, et dont le pas de temps est annuel, l'un des plans des agents agriculteur consiste à choisir un type de production. La durée de ce plan est très courte par rapport au pas de temps choisi et le plan est donc considéré comme instantané.

```
norm share_information intention:share_information
threshold:thresholdNorm instantaneous: true{
...
}
```

Code 5.14 – Exemple de définition d'une norme dans GAMA

5.3.4 Exécution de l'architecture

Le fait de définir le comportement d'un agent selon l'architecture BEN dans GAMA, en indiquant que l'agent suit l'architecture de contrôle *simple_bdi*, permet de donner un ordre d'exécution aux différents processus définis par le modélisateur. Cette exécution suit l'architecture théorique exposée dans le chapitre 4 mais son implémentation pratique possède quelques spécificités explicitées dans cette section.

La première étape de l'implémentation est l'initialisation du comportement des agents. En utilisant l'architecture BEN, un agent bénéficie de plusieurs variables parmi lesquels les bases de données contenant les connaissances de l'agent, les variables représentant la personnalité de l'agent ainsi que diverses variables indiquant quelles parties de l'architecture utiliser et selon quelles conditions :

— **probabilistic_choice** : variable booléenne indiquant si la prise de décision se fait de façon déterministe (avec une valeur à faux) ou de façon probabiliste (avec une valeur à vrai). Par défaut à faux.

— **use_emotions_architecture** : variable booléenne permettant d'utiliser le moteur émotionnel, créant automatiquement des émotions en fonction des connaissances de l'agent. Par défaut à faux.

— **use_social_architecture** : variable booléenne permettant d'utiliser le moteur social afin de mettre à jour les relations sociales de l'agent. Par défaut à faux.

— **use_personnality** : variable booléenne permettant d'utiliser les variables représentant la personnalité dans la création des émotions, la mise à jour de la personnalité et l'initialisation de toutes les variables comportementales de l'agent comme son obéissance, son charisme ou sa réceptivité émotionnelle.

— **use_persistence** : variable booléenne permettant de modifier les probabilité de persistance des intentions et des plans, indépendamment des valeurs de personnalité.

— **persistence_coefficient_intentions** : valeur réelle entre 0 et 1 indiquant la probabilité de conserver son intention courante à chaque pas de temps, indépendamment des modifications des connaissances de l'agent.

— **persistence_coefficient_plans** : valeur réelle entre 0 et 1 indiquant la probabilité de conserver son plan en cours, indépendamment des modifications des connaissances de l'agent.

La définition d'une personnalité, via la variable **use_personality** permet de donner des valeurs aux cinq variables représentant les cinq dimensions de la personnalité, comme expliquée en section 3.2.1. Dans *simple_bdi*, les cinq variables de la personnalité sont : **openness**, **conscientiousness**, **extroversion**, **agreeableness** et **neurotism**. Par défaut, ces variables sont initialisées à 0.5 et elles jouent un rôle dans la création des émotions, dans la contagion émotionnelle, la mise à jour des relations sociales et la gestion du système normatif. Le modélisateur est alors libre d'utiliser ces variables dans la cognition, à travers la perception ou les conditions d'activation d'un plan.

Une fois l'initialisation terminée, l'architecture comportementale est activée dès le premier pas de temps. Une fois toutes les actions réflexes effectuées (définies à travers le *statement* **reflex**), l'architecture exécute l'ensemble des *statements* **perceive** définis. Chaque perception activée exécute alors son bloc de code dans l'ordre dans lequel il est défini. Il n'y a donc pas d'ordre prioritaire, a priori, entre la mise à jour des connaissances, la contagion émotionnelle, la création de relations sociales ou le contrôle des normes. De même, si plusieurs perceptions sont définies et sont activables dans le contexte en cours, ces perceptions sont exécutées les unes après les autres, dans l'ordre dans lequel elles ont été définies.

L'exécution du deuxième module de l'architecture BEN se fait de deux façons. Les processus automatiques de création des émotions et de mise à jour des relations sociales sont exécutés de façon implicite ; les émotions sont créées lors de l'ajout d'une croyance ou d'une incertitude, notamment par le biais du *statement* **focus** alors que les relations sociales sont mises à jour lors de la perception, à travers le *statement* **socialize**. Les règles d'inférences et les lois, par contre, sont

explicitement exécutées après la perception, là encore dans l'ordre dans lequel elles sont définies si plusieurs d'entre-elles sont activables au moment concerné.

L'architecture exécute ensuite le moteur normatif puis le moteur cognitif si aucune décision n'a été prise par le biais des obligations. Dans le cas d'une intention venant des désirs, les normes sont évaluées avant les plans d'action. Ainsi, si une norme n'est pas choisie alors qu'elle pouvait l'être dans le contexte en cours, c'est uniquement parce que l'agent a délibérément choisi d'y désobéir.

Enfin, le quatrième module d'évolution des connaissances diminue la durée de vie des états mentaux cognitifs, réduit l'intensité émotionnelle et met à jour le statut des normes, pour faciliter le contrôle normatif par un autre agent comme expliqué en section 4.2.4. GAMA découpe le temps de simulation en pas de temps. Pour chaque pas de temps, chaque agent encore dans la simulation est activé et exécute son comportement. Ainsi, la durée de vie des états mentaux cognitifs se définie par un nombre de pas de temps avant que l'état mental ne disparaisse des bases de l'agent. L'étape de diminution de cette durée de vie revient à faire réduire la durée de vie de 1 à chaque pas de temps.

5.4 Discussion et conclusion

Dans cette section, l'ensemble de l'implémentation de l'architecture BEN est discutée. Le but est d'expliquer les choix réalisés pour transformer l'architecture théorique présentée dans le chapitre 4 en une architecture utilisable dans une plateforme de simulation, afin de définir des comportements agents avec des dimensions cognitives, affectives et sociales.

5.4.1 Discussion

L'implémentation de l'architecture BEN dans GAMA a été faite en essayant de rendre son fonctionnement le plus simple possible pour le plus grand nombre. Pour cela, l'architecture implémentée s'est inspirée des principes de programmation de GAMA et de son langage, le GAML, qui ont prouvé leur capacité à être utilisé même par des modélisateurs n'étant pas des informaticiens [MB14] [RRO15]. La partie cognitive de BEN a aussi montré qu'elle était utilisable par un public novice sur de telles architectures [TBC+17] mais aussi qu'elle n'était pas plus compliqué à utiliser qu'une machine à états finis, tout en offrant plus de profondeur dans la définition du comportement [ATDG17].

Chaque connaissance mise en jeu dans BEN possède dans GAMA son propre type de donnée, au plus près du formalisme proposé dans le chapitre 3. Ainsi, le modélisateur peut manipuler dans GAMA les mêmes types de hauts niveaux que ceux de l'architecture théorique, facilitant le passage de la théorie à la pratique.

Pour les processus mis en jeu dans l'architecture BEN, chaque processus modifiable par l'utilisateur possède son propre statement. Là encore, le but est de faciliter le passage d'un modèle théorique à un modèle pratique, simplement en traduisant les concepts dans le langage de programmation de la plateforme. Pour les processus automatiques, ils sont exécutés implicitement, sans que l'utilisateur n'ait besoin de les configurer, là encore facilitant l'utilisant de concepts de hauts niveaux qui demanderaient beaucoup de temps à implémenter de zéro.

Même si ce chapitre présente en détail l'ensemble des processus implémentés dans *simple_bdi*, le modélisateur a toujours le choix de n'utiliser que des parties de l'architecture BEN, conformément à sa définition modulaire. Pour cela, il suffit d'utiliser les variables définies en section 5.3.4 comme des boutons marche/arrêt, afin de n'inclure que les parties nécessaires au comportement de l'agent dans la situation étudiée. De même, pour chaque processus, le modélisateur n'est pas obligé de tous les utiliser, ne définissant que ceux jugés nécessaires dans le modèle comportemental.

En plus des *statements* présentés dans ce chapitre, et permettant de définir des portions du comportement de l'agent, de nombreuses fonctions et actions ont été développées pour mani-

puler les connaissances de l'agent manuellement avec plus de finesse, pour les utilisateurs plus expérimentés. Des actions *get*, *has* et *set* permettent de récupérer les connaissances de l'agent, de tester leur présence dans les bases de l'agent ou bien même de modifier directement et manuellement des champs particuliers de ces connaissances.

La profusion de ces fonctions et actions a pour but de s'adapter le plus possible aux modélisateurs. Il est donc possible de réaliser des actions similaires de différentes façons, en fonction des habitudes de chacun. Par exemple, une croyance peut être ajoutée avec un *focus*, mais peut aussi être ajoutée manuellement avec un *add_belief()* dans un *statement ask myself* comme le montre le code 5.7. Il est aussi possible de se passer de règles d'inférences en écrivant manuellement, lors de la perception par exemple, des règles "si-alors" permettant de modifier les états mentaux cognitifs d'un agent en fonction de ses connaissances précédemment acquises. Ainsi, l'architecture s'adapte au modélisateur et non l'inverse.

Sur le plan du temps de calcul, chaque processus, à part la création d'émotions, consiste dans le test d'appartenance d'une valeur à une liste. Le temps de calcul dépend donc de la longueur de cette liste, correspondant à une base d'états mentaux cognitifs, la base des émotions ou la base des relations sociales. Seule la création des émotions implique un double test d'appartenance effectué sur deux bases (soit d'état mentaux cognitif, soit d'émotions, soit de relations sociales). Les calculs effectués sont donc simples et en temps linéaire en moyenne.

La plateforme GAMA disposant de possibilités de calculs parallèles, l'implémentation de l'architecture BEN a aussi été parallélisée [TBDV17]. Cette parallélisation consiste à calculer en parallèle certains éléments du comportement des agents, comme la perception de l'environnement, l'exécution des règles d'inférence ou encore l'évolution des connaissances après l'exécution du plan. Seuls les éléments sociaux, nécessitant l'accès aux autres agents, ainsi que la prise de décision et l'exécution du plan, modifiant l'environnement partagé par tous les agents, ne sont pas parallélisés. En activant l'implémentation parallèle de l'architecture, via la définition de l'architecture de contrôle **parallel_bdi**, le modélisateur peut utiliser des processeurs de calcul multi-cœur afin d'accélérer la prise de décision de ses agents, sans pour autant impacter le comportement de la simulation.

5.4.2 Conclusion

Ce chapitre montre l'implémentation de l'architecture BEN, développée et détaillée dans le chapitre 4, effectuée dans la plateforme de simulation GAMA. Cette implémentation permet de définir des agents ayant des compétences cognitives, affectives et sociales, dans une simulation à base d'agents. Cette implémentation conserve le coté modulaire de l'architecture, en proposant au modélisateur de nombreuses fonctions et actions permettant une grande adaptation en fonction du niveau en programmation de l'utilisateur.

L'architecture BEN peut être implémentée dans d'autres plateformes de simulations et avec d'autres objectifs. Ici, nous avons fait le choix de privilégier la facilité d'utilisation et la simplicité de calcul, par exemple en incluant le statut du système normatif (quelles normes sont activables, quelles normes sont violées et quelles normes sont suivies) directement dans chaque norme stockée par l'agent. Dans une implémentation de l'architecture BEN, l'important est de conserver le découpage en 4 modules principaux, chacun composé de plusieurs processus. Si l'ordre de ces 4 modules (perception, gestion des connaissance, prise de décision, dynamisme des connaissances) est impératif, l'ordre d'exécution de chaque processus peut évoluer en fonction des implémentations.

L'implémentation présentée dans ce chapitre est d'ores et déjà disponible dans GAMA et peut être utilisée par n'importe qui désirant ajouter des composantes cognitives, affectives ou sociale à ses agents.

Chapitre 6

Application de BEN à l'évacuation de bâtiments

Sommaire

Ce chapitre présente deux exemples d'application de l'architecture BEN. Le but est de montrer comment BEN peut être utilisé pour modéliser et simuler des acteurs humains sur des cas d'étude réels.

Les deux cas étudiés ci-après portent sur l'évacuation de bâtiments par une foule humaine dans le contexte d'un incendie avec dégagement de fumées toxiques. Dans un premier temps, nous présentons le cas de l'incendie de la boite de nuit Kiss Nightclub au Brésil pour lequel nous avons développé un modèle pour simuler le comportement des personnes fuyant le danger. Ce modèle est ensuite transposé sur le second cas d'étude, portant sur l'évacuation de la boite de nuit Station Nightclub aux États-Unis. Le but de ce deuxième cas d'étude est de montrer que le comportement développé sur le premier cas d'étude est suffisamment générique pour être utilisé pour un autre cas avec un contexte différent.

Ces exemples permettent de montrer comment les dimensions cognitives, affectives et sociales de BEN peuvent être utilisées pour définir le comportement d'agents évacuant un bâtiment, offrant un comportement crédible qui conserve une explication de haut niveau. Les dimensions cognitives, affectives et sociales sont utilisées de la façon suivante :

— Dimensions cognitives : chaque agent va prendre une décision en fonction de sa perception, de ses croyances et de ses désirs. Différents plans d'action permettent à l'agent de changer d'action en fonction du contexte.

— Dimensions affectives : chaque agent va réagir en fonction de l'intensité de sa peur par rapport à un possible incendie, les amenant à évacuer sans avoir forcément de croyance sur l'incendie. Chaque agent dispose d'une personnalité qui paramètre les différents aspects de son comportement.

— Dimensions sociales : les agents vont réagir les uns en fonction des autres, par l'intermédiaire de la contagion émotionnelle, de relations sociales et d'un système de normes sociales. Les agents devront aussi suivre les consignes officielles d'évacuations modélisées par des lois.

Ces cas d'étude permettent aussi de montrer que BEN permet de simuler des centaines d'agents ayant des capacités cognitives, affectives et sociales dans un temps acceptable. En effet, l'un des objectifs de cette architecture étant de ne pas trop alourdir le temps de calcul des simulations, le test sur un exemple de plusieurs centaines d'agents permet d'avoir un retour précis sur ce point.

Enfin, ces deux exemples cherchent à montrer que BEN permet de définir facilement des comportements complexes, mais aussi que son implémentation dans GAMA permet de facilement passer du modèle théorique de comportement à des simulations. Pour cela, nous présentons la méthodologie utilisée pour modéliser le système étudié ainsi que des extraits de code.

6.1 Évacuation de la boite de nuit Kiss Nightclub au Brésil

La gestion de crise lors de l'évacuation de bâtiments est un problème qui a été souvent étudié par la simulation sociale [SA04]. Il s'agit en effet de cas qui ont souvent été bien documentés et détaillés par le biais de jugements et de rapports d'experts. De plus, ces évacuations correspondent à des situations où l' environnement est bien délimité, connu, et où les personnes à simuler doivent prendre des décisions dans un court laps de temps dans des conditions de stress élevé. C'est aussi une situation qui se prête bien à l'utilisation de l'architecture BEN puisque les acteurs simulés prennent des décisions de façon cognitive et affective mais aussi de façon sociale (phénomène de foule).

Dans la suite de cette section, nous présentons le contexte général de l'évacuation du Kiss Nightclub et le modèle de comportement des agents représentant les personnes fuyant l'établissement qui utilise l'architecture comportementale BEN. Nous proposons également, dans une dernière partie, une discussion sur l'utilisation de BEN.

6.1.1 Présentation de la simulation

Dans cette sous-section, le premier cas d'exemple sur lequel est utilisé BEN pour simuler une situation réelle est introduit. Une première partie présente les informations générales sur la situation étudiée alors qu'une seconde partie explique les premières phases de la modélisation, permettant de simuler le bâtiment, l'incendie et la propagation de la fumée.

Introduction au cas d'étude

Le 27 Janvier 2013, un incendie s'est déclenché dans le boite de nuit Kiss Nighclub à Santa Maria, dans l'état de Rio Grande du Sud au Brésil, causant la mort de 242 personnes. Cette nuit là, un groupe de musique a ponctué sa prestation sur scène en allumant un feu d'artifice à l'intérieur du club. Ce feu d'artifice a mis le feu au plafond, dégageant des fumées nocives qui se sont répandues dans l'ensemble de l'établissement.

Lors de l'enquête des autorités, plusieurs éléments ont été mis en évidence comme des facteurs aggravant de cette tragédie : entre 1200 et 1400 personnes étaient présentes à l'intérieur de la boite de nuit qui ne pouvait contenir officiellement que 641 personnes, il n'y avait qu'une seule porte de sortie qui servait aussi de porte d'entrée, il n'y avait pas d'alarme incendie ni de détecteur de fumée, il n'y avait pas non plus d'installation fixe d'extinction automatique à eau et enfin, les panneaux indiquant la sortie étaient cassés. Les seuls panneaux lumineux présents dans l'établissement indiquaient la direction des toilettes. L'enquête a aussi mis en évidence que la plupart des décès ont eu pour cause l'asphyxie et sont survenus aux abords des toilettes [Ati13].

Cette tragédie a été étudiée précédemment par d'autres chercheurs [MAS+17] qui ont cherché à déterminer si le fait de suivre les règles officielles brésiliennes en matière de sécurité aurait permis de réduire le bilan humain. Le modèle de comportement développé dans ces recherches est assez simple avec des agents réactifs qui décident de fuir au moment où le feu est déclenché. Les agents sont ensuite ralentis dans leur fuite par les meubles, principalement des tables et des chaises, placés aléatoirement dans le club. Une fois rattrapés par le feu, les agents sont déclarés

décédés. Les auteurs comparent alors la situation réelle à d'hypothétiques cas où plusieurs sorties auraient été disponibles. Le nombre d'agents présents au lancement de la simulation est aussi modifié pour voir si un nombre raisonnable de personnes dans cette boite de nuit aurait permis d'éviter le drame.

L'approche que nous proposons pour simuler cet accident est différente. En effet, notre objectif est de recréer le comportement des personnes prises dans cette tragédie de la façon la plus crédible possible, c'est à dire obtenir un comportement proche d'un comportement humain dans une telle situation.

Structure du modèle

La figure 6.1 montre le diagramme de classe que nous proposons pour représenter les différents composants intégrés dans la simulation : les agents "personnes" simulant les acteurs humains sont localisés dans l'environnement qui est composée d'un quadrillage de cellules pouvant représenter les murs, la sortie, l'incendie ou les toilettes.

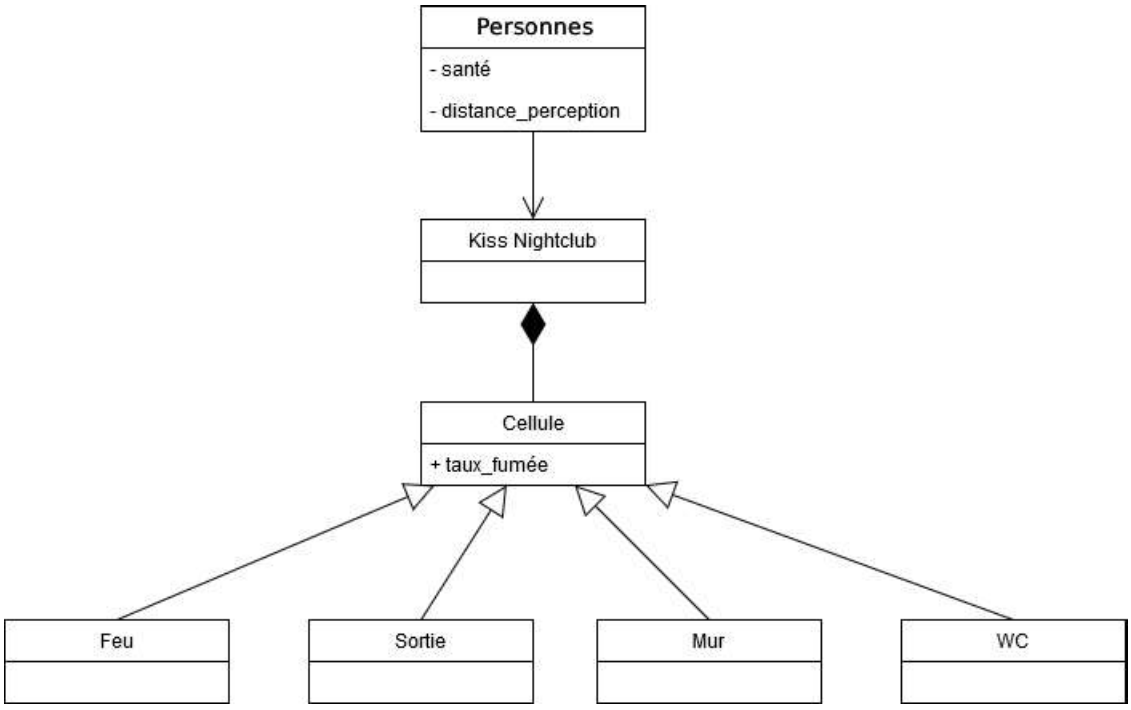

FIGURE 6.1 – Diagramme de classe du modèle d'évacuation du Kiss Nightclub

L'environnement du Kiss Nightclub a été recréé numériquement à partir des plans du club présentés dans [Ati13]. La figure 6.2 présente cet environnement : les triangles noirs à fond blanc représentent les agents cherchant à évacuer, les traits épais noirs les cloisons ou les barrières trop hautes pour permettre le déplacement des personnes, la zone bleue en bas la porte d'entrée/sortie et enfin, le disque jaune dans le coin en haut à gauche le point de départ de l'incendie. Ce point de départ du feu a été placé sur la scène, approximativement là où le vrai départ de feu a eu lieu.

Le placement précis des personnes la nuit de l'incident n'est pas connu. Néanmoins, sachant que la boite de nuit était surpeuplée, une hypothèse qui peut être faite est que la population à l'intérieur du bâtiment était répartie sur l'ensemble de l'établissement. Ainsi, nous avons fait le choix, au lancement de la simulation, de placer aléatoirement dans le plan de la boite de nuit les agents simulant les acteurs humains.

De façon à permettre une prise en compte fine des décisions et actions des différents acteurs, nous avons fait le choix de prendre des pas de temps correspondant à 1 seconde. A chaque pas de temps, deux principales dynamiques sont activées : les actions des acteurs et la propagation de la fumée.

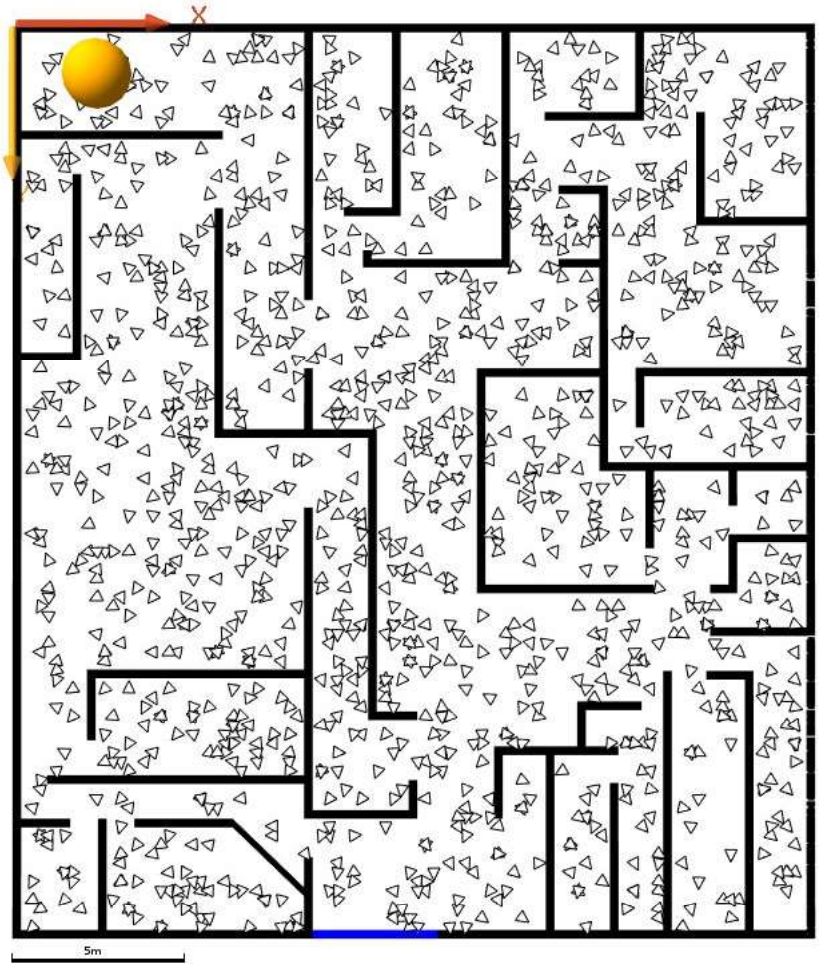

FIGURE 6.2 – Reproduction du Kiss Nighclub au moment de l'incendie, avec 1300 agents placés aléatoirement

En effet, la fumée dégagée par le feu a été à l'origine de la très grande majorité des décès. Ainsi, c'est la propagation de la fumée à l'intérieur du bâtiment qui nous intéresse plus que l'évolution de l'incendie en lui-même. Le déplacement de la fumée dans la boite de nuit a été modélisé en suivant une étude réalisée par le gouvernement français sur la dangerosité des incendies et de leurs fumées [CC05]. L'idée principale est que la fumée se diffuse à vitesse constante depuis son point d'émission et est dangereuse lorsqu'elle dépasse un certain seuil de concentration dans l'air. Pour autant, cette vitesse et cette dangerosité sont variables selon les matériaux en jeu dans la combustion. Dans le cas du Kiss Nightclub, et en absence d'informations précises, il a été décidé de modéliser une fumée moyenne ce qui signifie que l'établissement est rempli en 4 à 5 minutes et que les personnes perdent connaissance après une exposition de plusieurs dizaines de secondes à des taux de concentration importants. La figure 6.3 montre le résultat visuel de la propagation de la fumée depuis le point de départ du feu après 2 minutes de temps simulé. Les différents niveau de gris indiquent le niveau de concentration de la fumée, les zones blanches n'ayant pas du tout de fumée alors que les zones plus foncées ont une concentration dangereuse de fumée.

Le sol de la boite de nuit est divisée selon un quadrillage, observable sur la figure 6.3 en lignes blanches dans les zones de forte fumée en fond noir. Chaque case de ce quadrillage possède un pourcentage indiquant la proportion de fumée se trouvant dans le volume au-dessus d'elle même jusqu'au plafond. Si ce pourcentage est à 0, cela signifie qu'il n'y a pas de fumée dans le volume au-dessus de la case, si ce pourcentage est à 100, cela signifie que le volume au-dessus de la case est entièrement rempli de fumée.

Chaque agent simulant un acteur humain possède un entier représentant son énergie, initialisé à 100; cette valeur va être amenée à descendre à cause de la fumée respirée. Dans le cas du

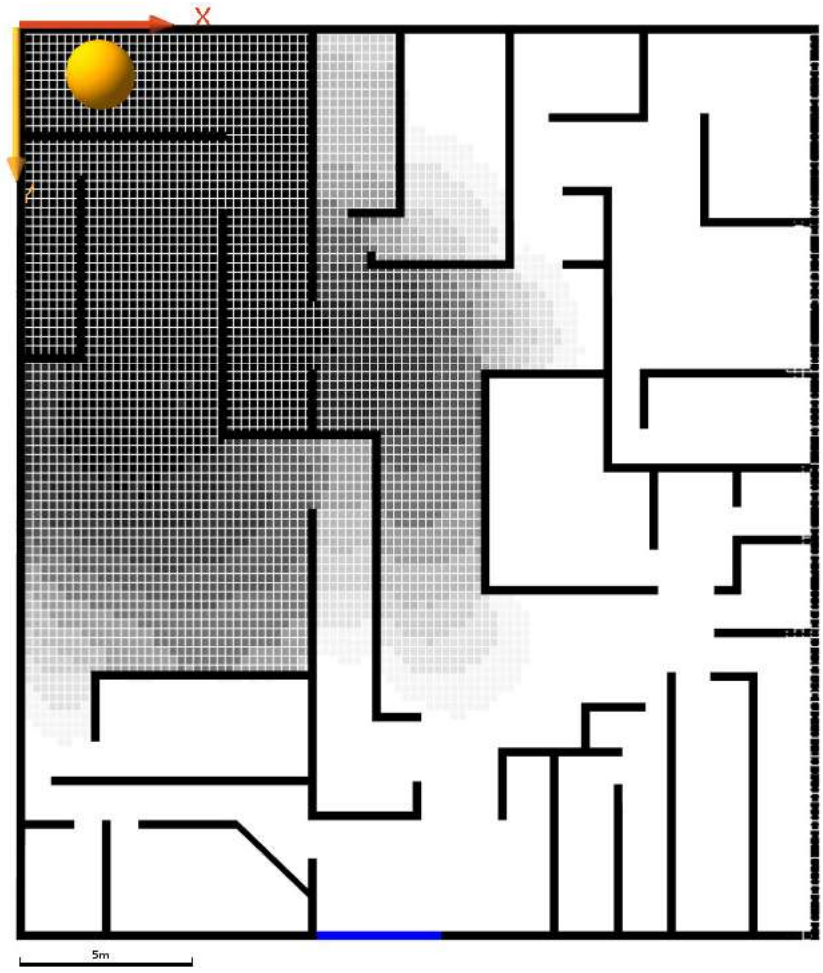

FIGURE 6.3 – Propagation de la fumée à travers le Kiss Nightclub après deux minutes de temps simulé

Kiss Nightclub, où la fumée est estimée dans la moyenne, une personne perd connaissance après être restée une cinquantaine de secondes dans la fumée épaisse. Ainsi, si l'agent se trouve sur une case avec plus de 50% de fumée, son énergie sera décrémentée de 2. Une fois l'énergie à 0, l'agent est considéré comme mort.

Nous détaillons dans la section suivante les comportement des agents humain simulés.

Le modèle complet se trouve au lien suivant : https ://github.com/mathieuBourgais/ExempleThese

6.1.2 Modélisation et implémentation du comportement des agents

Le point central de cet exemple est la modélisation du comportement des personnes devant évacuer le bâtiment. Dans le cas du Kiss Nightclub, leur comportement n'est pas connu avec précision, faute de témoignages suffisants. Il est ainsi nécessaire de faire des hypothèses sur les décisions prises par chacun.

Dans ce cadre, nous avons fait le choix d'avoir un comportement le plus générique possible afin qu'il puisse s'adapter à d'autre cas d'évacuation de bâtiments.

Les connaissances initiales de l'agent, au lancement de la simulation, sont de trois types : des croyances sur le monde qui l'entoure, des désirs initiaux ainsi que des relations sociales. À cela vient s'ajouter une personnalité, initialisée en suivant des distributions gaussiennes pour les 5 paramètres du modèle OCEAN présenté dans la section 3.2.1.

Plus précisément, l'ensemble des connaissances initiales de l'agent, relatif à l'évacuation de la boite de nuit, sont résumées et formalisées dans le tableau 6.1.

Les deux croyances sur l'emplacement de la porte ont des durées de vie différentes, représentant le fait que les agents oublient plus rapidement l'emplacement exact de la sortie que son

TABLEAU 6.1 – Connaissances initiales des agents évacuant le Kiss Nightclub

énoncé	formalisation	description
une croyance sur la position précise de la porte de sortie	$Belief_i(porteSortie, lifetime1)$	Chaque agent possède une croyance initiale sur la position précise de la porte de sortie avec une durée de vie de valeur $lifetime1$.
une croyance sur la position approximative de la porte de sortie	$Belief_i(directionSortie, lifetime2)$	Chaque agent possède une croyance initial sur l'emplacement approximatif de la porte de sortie, où $lifetime2 > lifetime1$.
un désir de s'amuser	$Desire_i(samuser, 1.0)$	Chaque agent souhaite s'amuser au lancement de la simulation et ce souhait a une priorité de 1.0.
un désir qu'il n'y ait pas d'incendie	$Desire_i(nonFeu, 1.0)$	Chaque agent souhaite qu'il n'y ait pas le feu dans la boite de nuit avec une priorité de 1.0. Ce désir ne peut pas devenir une intention.
un lien d'amitié avec un autre agent	$R_{i,j}(L, D, S, F, T)$	Chaque agent i possède possiblement un lien d'amitié avec un autre agent j.

emplacement approximatif. Ces durées sont initialisées avec une part d'aléatoire pour être différentes d'un agent à l'autre, modélisant le fait que certaines personnes ont une bonne mémoire et d'autres pas.

La simulation est initialisée à l'instant où l'incendie se déclenche dans la boite de nuit. au début de la simulation, les agents souhaitent passer une bonne soirée ; cela signifie à la fois s'amuser en passant du temps dans l'établissement mais aussi qu'il n'y ait pas de catastrophe. Ce deuxième souhait ne peut pas devenir une intention et déclencher de plan d'action pour l'atteindre. En revanche, le désir de s'amuser dans la boite de nuit est associé à un plan, une séquence d'actions élémentaires qui va, dans ce cas, correspondre à un déplacement aléatoire de l'agent dans un court rayon autour de lui.

Finalement, certains agents ont une relation sociale avec un autre agent choisi aléatoirement à l'initialisation de la simulation. Ce lien représente l'amitié entre ces deux personnes qui se connaissent avant la catastrophe et donc qui peuvent réagir différemment l'un envers l'autre. Ce lien social est initialisé aléatoirement en faisant en sorte que la valeur d'appréciation soit positive. L'ami avec lequel un lien social est partagé a lui aussi un lien social, qui ne sera pas nécessairement symétrique du premier.

Le code 6.1 présente l'implémentation des connaissances initiales des agents. Les prédicats représentant le monde du point de vue de l'agent sont déclarés en premier ainsi que la personnalité où chaque dimension est initialisée selon une distribution gaussienne. Ensuite, le *statement init* permet d'indiquer les actions que réalise l'agent au moment de l'initialisation de la simulation : l'ajout des désirs et des croyances ainsi que la création d'un lien d'amitié. À cela s'ajoute la définition d'un plan permettant de répondre à l'intention de s'amuser, représenter dans cet exemple par le prédicat $wandering$.

Ainsi, au lancement de la simulation, l'agent se déplace aléatoirement dans la boite de nuit,

```
species people control: simple_bdi{
predicate exitDoor <- new_predicate("exitLocation");
predicate directionExitDoor <- new_predicate("direction of the exit");
predicate fireSaw <- new_predicate("fire");
predicate noFire <- new_predicate("fire",false);
predicate wandering <- new_predicate("wander");

float openness <- gauss(0.5,0.12);
float conscientiousness <- gauss(0.5,0.12);
float extraversion <- gauss(0.5,0.12);
float agreeableness <- gauss(0.5,0.12);
float neurotism <- gauss(0.5,0.12);

init{
    do add_desire(wandering);
    do add_desire(noFire,1.0);
    do add_belief(predicate:exitDoor, lifetime: know_exitDoor_lt);
    do add_belief(predicate:directionExitDoor, lifetime:2*know_exitDoor_lt);
    if(!hasFriend and flip(propFriend)){
        friend <- one_of(people);
        hasFriend<-true;
        do add_social_link(...);
        ask friend{
            friend <- myself;
            hasFriend<-true;
            do add_social_link(...);
        }
    }
}

plan wander intention:wandering{
...
}

}
```

Code 6.1 – Initialisation des agents simulant les acteurs humains dans l'évacuation du Kiss Nightclub

suivant son désir de s'amuser. Ce comportement de base va être modifié par un changement dans l'environnement, induit par l'incendie.

Déclenchement de l'évacuation

C'est la modification des connaissances de l'agent qui va déclencher un changement de comportement ; si ses connaissances ne changent pas, l'agent n'a pas de raison de changer de comportement. Cette modification des connaissances se fait, dans un premier temps, via la perception de l'environnement, qui permet d'ajouter de nouvelles croyances ou incertitudes. Cette perception globale de l'environnement se ramène à plusieurs perceptions plus précises sur des éléments donnés du monde :

— La perception de la porte d'entrée/sortie permet de mettre à jour les croyances vis à vis de celle-ci.

— La perception du feu permet d'ajouter la croyance qu'il y a un incendie.

— La perception des cellules permet de récupérer des informations sur le niveau de fumée autour de l'agent.

— La perception des toilettes lorsque l'agent suit les panneaux de secours pour fuir permet d'acquérir la croyance que les panneaux de secours ne donnent pas la bonne information.

— La perception des autres agents simulant des acteurs humains permet de créer des relations sociales dynamiquement ainsi que de réaliser une contagion émotionnelle sur l'émotion de peur liée à l'incendie. De plus, si les agents perçus fuient, cela ajoute l'incertitude de l'incendie. Un contrôle d'une norme sociale visant à suivre les autres est aussi mise en place.

L'implémentation de ces perceptions se fait en utilisant les *statements* **perceive** et **focus** détaillés en section 5.3.1. Le code 6.2 montre des exemples d'implémentation de certaines de ces perceptions, ainsi que la définition de sanctions pour le contrôle de la norme.

```
perceive target:victim in:perceived_area{
    if(has_belief(fireSaw) and not myself.has_belief(fireSaw)){
        focus id:"fire" strength: uncertaintyConversion is_uncertain:true;
    }
    emotional_contagion emotion_detected:fearFire
        threshold:contagionThreshold;
    socialize trust:gauss(0.0,0.33);
    myself.perceivedOther<-self;
    enforcement norm: "followOthers"
        sanction: "trustSanction" reward: "trustReward";
}

sanction trustSanction{
    do change_trust(perceivedOther,-0.1);
}

sanction trustReward{
    do change_trust(perceivedOther,0.1);
}

perceive target:exit in:perceived_area{
    focus id:"exitLocation" lifetime: 20;
}

perceive target:fire in:perceived_area{
    focus id:"fire";
}

perceive target:bathroom in: perceived_area when: has_obligation(fleeing) {
    focus id:"wrongExit";
}
```

Code 6.2 – Exemple de définition de perceptions dans le cas du Kiss Nightclub

Une fois les perceptions effectuées, les connaissances de l'agent sont à jour par rapport à son environnement. Afin de modifier son comportement, l'agent doit modifier ses désirs, ses obligations et ses intentions. Pour ce faire, des règles d'inférence et des lois sont définies pour mettre à jour les désirs et obligations de l'agent en fonction de ses croyances et de ses incertitudes :

— Une loi qui crée l'obligation de suivre les panneaux de sortie de secours dès qu'il existe un doute suffisant (modélisé par un seuil d'obéissance) qu'il y a une catastrophe à fuir.

— Une règle d'inférence qui retire l'obligation de suivre les panneaux de sortie si l'agent a la croyance qu'ils donnent une mauvaise information.

— Une règle d'inférence qui retire l'intention et le désir de s'amuser et qui ajoute le désir de fuir si l'agent possède la croyance qu'il y a un incendie.

— Une règle d'inférence qui ajoute l'incertitude qu'il y a un incendie si l'agent possède la croyance qu'il est entouré de fumée.

— Une règle d'inférence qui retire l'intention et le désir de s'amuser et qui ajoute le désir de fuir si l'agent possède une émotion de peur liée à l'incendie

Le code 6.3 montre un exemple d'implémentation de certaines des règles d'inférence et de la loi nécessaires pour adapter les connaissances de l'agent en fonction de ses perceptions. Le seuil d'obéissance de la loi est initialisé à 1.0, indiquant que seuls les gens avec une obéissance maximale fuient dès qu'ils voient de la fumée en boite de nuit. Ce seuil est abaissé en fonction de la quantité de fumée entourant l'agent, jusqu'à atteindre 0.0 si l'agent est entouré par une grande quantité de fumée, ce qui implique que même les agents les plus réticents à la loi la suivent dans un cas d'urgence.

```
law belief: smokeSaw new_obligation:fleeing
    when: not has_belief(wrongExit) threshold:thresholdLaw;
rule belief: wrongExit remove_obligation: fleeing remove_intention:fleeing;
rule belief: fireSaw new_desire:fleeing
    remove_intention:wandering remove_desire:wandering ;
rule belief: smokeSaw new_uncertainty:fireSaw
    strength: smokeQuantity/100 when: not has_belief(fireSaw);
rule emotion:fearFire new_desire:fleeing
    remove_intention:wandering remove_desire:wandering threshold:riskAversion;
```

Code 6.3 – Exemple de définition de lois et de règles d'inférence dans le cas du Kiss Nightclub

Ces règles d'inférences permettent de déclencher le comportement de fuite de l'agent par rapport à ses perceptions de l'environnement. Par exemple, si un agent perçoit l'incendie, la croyance qu'il y a le feu lui est ajoutée, déclenchant ensuite un règle d'inférence qui va modifier ses intentions et ses désirs. Si l'agent possède la croyance qu'il est entouré de fumée, la création de l'incertitude qu'il y ait un incendie va créer, automatiquement grâce au moteur émotionnel, une émotion de peur liée à cet incendie. Lorsque l'intensité de cette émotion est suffisante, une règle est déclenchée pour créer le désir de fuir.

Différentes évacuations en fonction des conditions

Un partie importante du travail du modélisateur avec BEN consiste à définir des plans d'action et des normes permettant à l'agent de répondre à ses intentions. Dans le cadre de notre modèle d'évacuation, ces derniers vont majoritairement concerner le désir de fuir. Tous les agents auront le même désir de fuir l'incendie mais, en fonction du contexte dans lequel ils sont (s'ils connaissent l'emplacement de la sortie ou non, s'ils peuvent encore voir ou si la fumée est trop épaisse, etc), ils utiliseront des plans différents. Les plans et normes définis pour la simulation de l'évacuation du Kiss Nightclub sont détaillés dans le tableau 6.2.

Parmi ces moyens d'agir et de répondre à l'intention de fuir, deux sont des normes : le comportement qui amène à suivre un autre agent et celui qui répond à la loi. Comme l'indique le formalisme présenté en section 3.3.1, la définition d'une loi implique la définition d'une norme pour y répondre. Nous avons choisi d'implémenter le comportement consistant à suivre les autres agents sous la forme d'une norme, considérant ainsi ce comportement comme une norme sociale qui peut ne pas être suivie et amener à un contrôle déjà expliqué lors de la présentation des perceptions de l'agent.

TABLEAU 6.2 – Plans d'action et normes répondant à l'intention de fuir le Kiss Nightclub

conditions	actions	commentaires
l'agent a une bonne visibilité et possède la croyance sur l'emplacement précis de la porte de sortie	l'agent se dirige vers la sortie en courant	Dans ce plan, l'agent court vers la porte de sortie en suivant le chemin le plus court.
l'agent a une bonne visibilité et possède la croyance sur l'emplacement approximatif de la porte de sortie (il n'a plus de croyance sur l'emplacement précis de la sortie)	l'agent se dirige vers la sortie en courant	Dans ce plan, l'agent se déplace vers une cible à quelques mètres de lui dans la direction de la sortie. Le chemin n'est donc pas nécessairement le plus court.
l'agent a une bonne visibilité et n'a plus de croyances sur la porte de sortie	l'agent s'éloigne du danger	Dans ce plan, l'agent se dirige vers le point le plus loin de la fumée par rapport à lui-même.
l'agent a une bonne visibilité et n'a plus de croyances sur la porte de sortie	l'agent suit l'agent dans son champ de vision en qui il a la plus grande confiance	Dans cette norme, l'agent suit un autre agent en qui il a confiance.
l'agent a une mauvaise visibilité et possède la croyance sur l'emplacement précis de la porte de sortie	l'agent se dirige vers la sortie	Dans ce plan, l'agent se dirige vers la porte de sortie en suivant le chemin le plus court à une vitesse supérieur à la marche et inférieur à la course.
l'agent a une mauvaise visibilité et possède l'obligation de suivre les panneaux	l'agent se dirige vers les toilettes	Dans cette norme, l'agent se conforme à la loi lui indiquant de suivre les panneaux de secours.
l'agent a une mauvaise visibilité et possède la croyance que les panneaux de secours sont erronés	l'agent se déplace au hasard	Dans ce plan, l'agent se déplace aléatoirement dans la fumée.

La définition de ces plans et normes permet à l'agent d'agir dans tous les cas possibles auxquels il est confronté dans la fuite du Kiss Nightclub. De plus, en renseignant des conditions de fin à ces plans, l'agent a la possibilité de changer d'action lors d'un changement de contexte. Ainsi, si un agent a suivi les panneaux de secours, il ne va pas rester bloqué aux toilettes une fois qu'il aura la croyance que les panneaux sont erronés. L'architecture sélectionnera alors un nouveau plan en fonction du contexte de l'agent.

Comportement d'aide envers un ami

Les agents possédant des relations sociales d'amitié vont réagir si leurs amis ont besoin d'aide. Ainsi, si un agent A est ami avec un agent B et que ce dernier se trouve dans la fumée et ne sait pas où aller, l'agent A essayera d'aider l'agent B en traversant la fumée pour lui indiquer la direction de la sortie, si celui-ci la connaît. Ce comportement d'aide envers un ami implique la définition d'une perception pour créer la croyance que l'ami a besoin d'aide, d'une règle d'inférence pour arrêter la fuite et aider son ami et enfin d'un plan pour effectivement réaliser cette aide.

Une variable booléenne est ajoutée à chaque agent, indiquant si celui-ci a besoin d'aide ou non. Cette variable est initialisée à faux et devient vraie lorsqu'un agent perçoit de la fumée épaisse autour de lui sans avoir de croyances sur l'emplacement de la sortie. Une perception est alors définie, permettant d'ajouter la croyance qu'un ami a besoin d'aide lorsque l'agent ayant une relation

sociale d'amitié possède sa variable d'aide à vrai, si cet ami est à une distance inférieure à 5 mètres. Cette distance de perception modélise le fait que l'appel à l'aide d'une personne n'est entendu que dans un certains rayon et non dans l'ensemble de la boite de nuit, dû au bruit ambiant qui y règne.

Une règle d'inférence est définie, ajoutant à l'agent le désir d'aider son ami s'il a la croyance que ce dernier a besoin d'aide. Une condition d'activation est ajoutée à la règle, ne l'exécutant que si l'agent n'a lui-même pas besoin d'aide et s'il possède une valeur suffisante en agréabilité et en solidarité avec son ami. De plus, la règle va retirer l'intention de fuir et créer le désir d'aider son ami avec une durée de vie de 30 secondes. De cette façon, l'agent va reconsidérer son action de fuite pour aider son ami mais, s'il ne le trouve pas dans la fumée après 30 secondes, il reprendra sa fuite.

Finalement, un plan permettant de répondre à l'intention d'aider un ami est défini. Ce plan consiste à s'approcher de l'ami en question jusqu'à se trouver à une distance de moins d'un mètre et alors lui communiquer l'emplacement de la porte de sortie. Cette communication revient, en pratique, à ajouter une croyance portant sur le prédicat $porteSortie$ avec une durée de vie de 10 secondes, indiquant que cette information est précaire pour l'ami qui devra traverser la boite de nuit enfumée pour sortir.

6.1.3 Résultats et commentaires

La figure 6.4 représente un résultat visuel d'une simulation de l'incident du Kiss Nightclub, une minute et demi après le départ du feu. Les agents évacuant la boite de nuit sont représentés par des triangles dont la couleur indique le plan qu'ils suivent : les agents verts vont vers la sortie parce qu'ils connaissent son emplacement précis, les agents jaunes se dirigent en direction de la porte de sortie, les agents bleus suivent les panneaux lumineux, les agents rouges foncés sont perdus dans la fumée et ne croient plus les panneaux lumineux, les agents violets traversent la fumée car ils savent où se trouvent précisément la sortie, les agents marrons suivent un autre agent et les agents blancs ne fuient pas encore. Sur cette image, l'ensemble des comportements décrits en section 6.1.2 ne sont pas montrés, notamment le comportement d'aide envers un ami et le comportement qui consiste à fuir la fumée.

Comme expliqué en introduction de ce chapitre, Le but de cette expérimentation est double : montrer que l'architecture BEN permet de définir des comportements complexes et réalistes pour des agents simulant des humains et montrer que le temps de calcul d'une simulation utilisant BEN reste raisonnable, même pour des centaines d'agents. Ainsi, les résultats sont analysés par rapport à ces deux buts.

Résultats de l'expérimentation

Le nombre précis de personnes dans la boite de nuit au moment du drame n'est pas connu. Pour autant, les différents rapports s'accordent sur une estimation entre 1200 et 1400 personnes [Ati13]. Afin de couvrir l'ensemble de ces possibilités, des simulations ont été effectuées pour 1200, 1300 et 1400 agents au lancement de la simulation. Dans chaque cas, 10 simulations ont été effectuées en regardant à chaque fois le nombre d'agents considérés comme décédés, c'est à dire dont la jauge de vie a atteint 0. Les simulations ont été réalisées sur un ordinateur équipé d'un processeur intel I7 et de 16 Go de RAM.

Le résultat des simulations est montré dans le tableau 6.3. Pour chaque cas, un nombre moyen d'agents décédés ainsi qu'un écart type de cette donnée ont été calculés. Les cinq paramètres OCEAN de la personnalité ont été initialisés, pour tous les agents, par une distribution gaussienne centrée en 0.5 et avec un écart-type à 0.12, de sorte à ce que la valeur soit entre 0 et 1. La zone de perception de chaque agent est calculée à partir de valeurs réelles de distance de vue et de champs de vision.

Le seul paramètre initialement choisi dans le modèle est la quantité de fumée nécessaire pour être prise en compte ainsi que la quantité à partir de laquelle le champ de vision de l'agent est réduit. Ces deux paramètres ont été placés à 30 pour cent et 60 pour cent de fumée dans le champ

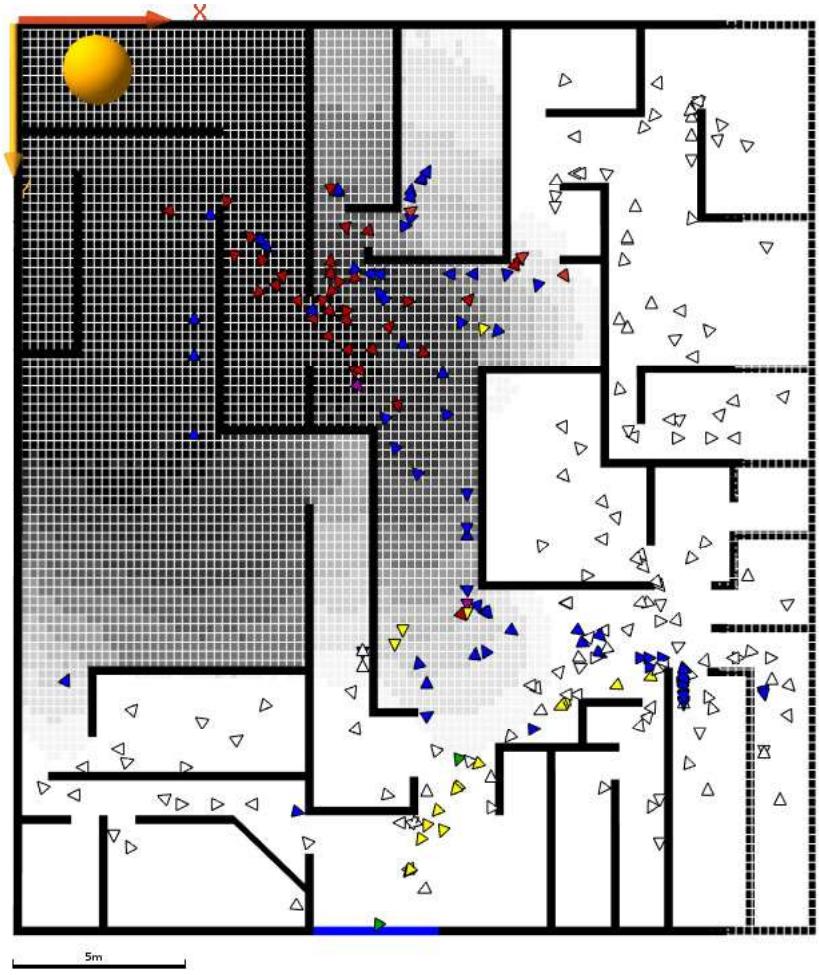

FIGURE 6.4 – Simulation de l'évacuation du Kiss Nightclub

de vision, indiquant d'abord une fumée faible mais en quantité trop importante pour être due à une machine de divertissement, puis une fumée noire en quantité suffisante pour indiquer un incendie.

TABLEAU 6.3 – Nombre d'agents morts lors de la simulation de la tragédie du Kiss Nighclub

nombre d'agents	1200	1300	1400
moyenne	230.2	237.7	249.4
écart type	20.1	15.6	32.6

Le tableau 6.3 montre que les résultats obtenus sont statistiquement proches du cas réel où il y a eu 242 victimes. De plus, l'architecture BEN permet de définir des simulations offrant de riches comportement, comme le montre la figure 6.4 où des agents dans des situations très similaires ont des comportements différents, symbolisés par des couleurs différentes. Ces différences de comportement s'expliquent par les personnalités différentes, créant ainsi des émotions d'intensité variables, des valeurs d'obéissance qui ne sont pas les mêmes, et ainsi de suite.

Une vidéo de la simulation de l'évacuation du Kiss Nightclub est disponible à l'adresse https://github.com/mathieuBourgais/ExempleThese. Sur cette vidéo, différents motifs comportementaux sont observés : les agents qui perçoivent beaucoup de personnes fuir fuient, une première vague de personnes quittent l'établissement dès le début parce qu'ils ont perçu l'incendie pendant que les autres continuent de s'amuser, les agents qui perçoivent la fumée tardivement dans la simulation ne se souviennent plus de l'emplacement de la sortie et suivent immédiatement les panneaux de secours ou encore le fait que des agents qui se déplaçaient au hasard dans la fumée finissent par

percevoir la sortie et s'échapper. Ces motifs comportementaux, couplés aux résultats statistiques et aux observations faites sur la simulation mènent à penser que le modèle développé est validé [ARB+06].

Le tableau 6.4 présente les temps moyens d'exécution pour chaque étape de calcul dans les différents cas simulés. Cela signifie que, pour 1200 agents initialement dans la boite de nuit, il faut 860 ms en moyenne pour calculer le comportement de tous les agents encore vivants dans l'établissement pour un pas de temps, représentant 1 seconde en temps simulé. Ce résultat, obtenu en utilisant la version parallélisée de l'architecture, montre que BEN est utilisable dans un temps de calcul raisonnable, compte tenu du fait que des calculs lourds comme les calculs de plus courts chemins sont pris en compte dans les résultats du tableau 6.4.

TABLEAU 6.4 – Temps de calcul, en millisecondes, obtenus pour la simulation de la tragédie du Kiss Nighclub

nombre d'agent	1200	1300	1400
moyenne	861.7	841.4	884.9
écart type	78.4	57.7	101.2

Ces résultats tendent à montrer que les objectifs fixés lors de la réalisation de l'architecture BEN ont été atteints sur un cas réel d'évacuation de bâtiment en feu. Tout d'abord, BEN a permis de modéliser un comportement complexe pour les agents simulant des acteurs humains, en incluant la cognition mais aussi les dimensions affectives et sociales du comportement. Ces comportements complexes restent néanmoins simples à paramétrer puisqu'ils sont liés aux cinq valeurs de la personnalité, qui représentent les seuls variables comportementales à renseigner.

Cette simulation du Kiss Nightclub montre aussi que la définition d'un comportement plus complexe permet d'obtenir un résultat crédible, que ce soit dans le nombre de décès ou bien dans la visualisation de l'évacuation, avec des agents qui ne fuient que lorsqu'ils ont conscience d'un danger, chacun réagissant en fonction de sa personnalité et de son contexte social.

Commentaires sur la modélisation

Comme expliqué en section 6.1.2, le comportement des agents est fondé sur des hypothèses. Par exemple, aucun agent "sauveteur" qui aiderait les autres à sortir, indépendamment de leurs relations sociales, n'a été intégré. De même, aucun agent ne panique au point de rester sur place ou de courir au hasard dans l'établissement. Ainsi, seuls les comportements contrôlables et mesurables ont été conservés dans le modèle, afin de maîtriser la simulation.

Toujours dans les comportements qui ne sont pas pris en compte dans la modélisation, les agents ne peuvent pas retourner dans la boite de nuit. Une fois qu'un agent a quitté la boite de nuit, c'est à dire qu'il a atteint la porte de sortie de l'établissement, il est retiré de la simulation. Ainsi, il n'est plus pris en compte dans les étapes de calcul et ne peut plus agir sur la simulation. Là encore, il est possible d'imaginer que des personnes sont retournées dans la boite de nuit pour aider des gens à sortir de la fumée mais ce nombre n'est pas mesurable ni contrôlable ; nous avons donc fait le choix de ne pas inclure ce comportement dans le modèle.

L'étude qui avait été précédemment menée sur ce cas du Kiss Nightclub [MAS+17] prenait en compte les meubles à l'intérieur de l'établissement dans le déplacement des agents. L'hypothèse évoquée est que les tables et les chaises ont réduit la vitesse de déplacement des personnes évacuant la boite de nuit. Dans la modélisation proposée dans cette section, les meubles n'ont pas été pris en compte car il n'est pas possible de connaître leur emplacement précis au moment de l'incendie ni même l'impact exact qu'ils ont eu lors de l'évacuation. Cette dimension est tout de même prise en compte dans la vitesse d'évacuation des agents, qui fuient à une vitesse légèrement supérieur à une vitesse de marche, et inférieur à une vitesse de course.

Un autre choix pouvant être discuté dans la modélisation proposée dans cette section est le fait de ne donner qu'un seul ami au maximum à chaque agent. Là aussi, il n'y a pas de témoignages précis sur le nombre d'amis moyen par personne dans l'établissement au moment de l'incendie

ce qui impose de prendre l'hypothèse la plus faible. De plus, les quelques témoignages de rescapés [Ati13] indiquent que chacun a quitté l'établissement le plus rapidement possible, sans se coordonner longuement avec des amis.

Enfin, seuls les agents décédés, c'est à dire dont la jauge de vie a atteint zéro à cause de la fumée, ont été comptabilisés dans les résultats de la simulation : les blessés ne sont pas pris en compte. Dans le cas réel, de nombreuses personnes ont été blessées dans la bousculade liée à l'évacuation. Pour autant, ce paramètre est compliqué à reproduire et à contrôler en simulation, c'est pourquoi il n'a pas été inclut dans l'étude menée dans cette section.

Commentaires sur l'utilisation de BEN

Cet exemple montre que BEN permet de traduire directement les hypothèses sur le comportement des acteurs humains simulés en conservant une description de haut niveau. Ainsi, la prise de décision des agents repose sur des concepts généraux, permettant d'expliquer les résultats observés en termes de combinaisons d'éléments cognitifs, affectifs ou sociaux. Il est ainsi possible à tout instant et pour chaque agent de retracer son cheminement cognitif et de comprendre ce qui l'a amené à entreprendre son action en cours.

La modélisation proposée en section 6.1.2 n'est pas unique; il est possible de modéliser les mêmes comportements, ou des comportements similaires, de différentes façons, toujours en utilisant BEN. Par exemple, l'ajout de désirs en fonction de croyances réalisé à travers des règles d'inférence peut être implémenté directement lors de la perception : le modélisateur utilise l'action $add\_desire()$ à l'intérieur d'un *statement perceive*. Ceci montre que l'implémentation de l'architecture, ainsi que des statements, actions et opérateurs supplémentaires détaillés dans le chapitre 5, s'adapte à chaque modélisateur. En fonction des compétences en programmation de chacun, et de la façon de voir la décomposition du comportement, plusieurs implémentations sont possibles pour un même problème. Ainsi, l'architecture s'adapte à chacun et impose le moins possible de contraintes à la façon de programmer.

Malgré tout, il existe de bonnes pratiques, dans l'utilisation de BEN, qui ne sont pas obligatoires mais qui permettent d'utiliser pleinement et simplement les capacités de l'architecture tout en conservant sa capacité de description du comportement par des concepts de haut niveau. Parmi ces bonnes pratiques, il est conseillé de déclarer à l'avance les prédicats qui seront utilisés dans le reste du comportement, afin d'alléger le code et de le rendre plus lisible. Sur la décomposition du comportement, il est conseillé d'avoir des perceptions simples qui se concentrent sur les croyances et les incertitudes, sur la contagion émotionnelle, sur le contrôle des normes et sur la création des relations sociales. Ces perceptions sont alors complétées par des règles d'inférences et des lois, elles aussi chacune assez simples, puis par des plans et des normes qui peuvent être plus complexes. Cette décomposition du comportement suit la décomposition de l'architecture en modules, détaillée dans le chapitre 4, et permet de plus facilement comprendre les actions effectuées par les agents, notamment pendant la phase de développement du modèle. Utilisée ainsi, l'architecture permet de simplement modéliser puis simuler une situation.

L'exemple développé dans cette section montre aussi la modularité de BEN. Si la quasi totalité des processus définis par l'architecture sont utilisés dans le modèle, seule la mise à jour automatique des liens sociaux n'est pas à l'œuvre, tous les composants ne sont pas actifs en même temps sur l'ensemble des comportements. Par exemple, un agent voyant l'incendie et quittant la boite de nuit au début de l'incident n'utilisera ni ses émotions, ni ses relations sociales pour prendre une décision. Chaque agent utilise donc les processus qui lui sont nécessaires à un instant donné, sans avoir besoin de tous les activer.

Comme indiqué dans l'étude des résultats obtenus en section 6.1.3, les paramètres comportementaux de l'agent se résument aux cinq paramètres OCEAN de la personnalité. Ceci permet de faciliter la paramétrisation du modèle où seules cinq variables sont à renseigner pour le comportement de l'agent. Le reste des variables à initialiser sont des paramètres liés à l'environnement qui peuvent avoir une valeur fondée sur des données statistiques ou des études précises.

L'architecture BEN permet aussi de ramener la problématique de l'évacuation d'un bâtiment

à un seul désir de fuite qui, lorsqu'il sera choisi pour devenir une intention, pourra être traité de manières différentes en définissant de multiples plans et normes. Ainsi, le désir de fuite n'apparaît pas en même temps chez tous les agents ce qui signifie que leur contexte est différent au moment où ils doivent traiter leur intention de fuir, les amenant à choisir différents plans d'action. De plus, un plan sélectionné n'est pas nécessairement conservé jusqu'à la fin de la simulation. Lorsque le contexte évolue autour de l'agent, celui-ci va conserver son intention de fuir mais changer de plan utilisé pour y répondre. Une autre façon de faire aurait été de proposer plusieurs désirs spécifiques de fuite, puis d'associer à chaque désir un unique plan. Là encore, c'est l'architecture qui s'adapte au modélisateur et à sa conception du comportement des agents.

6.2 Évacuation de la boite de nuit Station Nightclub aux États-Unis

Un second cas d'utilisation de BEN sur une situation réelle est détaillé dans cette section. Il s'agit là encore de l'évacuation d'un bâtiment dans une situation d'incendie; comme au Kiss Nightclub au Brésil, l'évacuation du Station Nightclub a été une catastrophe, menant au décès de plusieurs personnes. L'intérêt d'un tel exemple est de reprendre le comportement développé pour une évacuation de bâtiment et de montrer que celui-ci est suffisamment générique pour être utilisé dans le cadre d'un autre cas d'étude.

6.2.1 Présentation du cas

Le but de cet exemple est de transposer le modèle comportemental proposé pour l'évacuation de la boite de nuit Kiss Nightclub sur un autre cas d'évacuation de bâtiment dans un contexte un peu différent. De cette façon, il peut être montré que le comportement proposé n'est pas lié à un cas précis mais qu'il peut être adapté simplement, c'est à dire sans changer fondamentalement sa description de haut niveau.

Introduction du cas d'étude

Le 20 Février 2003, à West Warwick dans le Rhode Island, aux États-Unis, la boite de nuit Station Nightclub a subi un incendie meurtrier. 465 personnes se trouvaient dans l'établissement au moment ou un groupe de musique a ponctué sa prestation par des feux d'artifices qui ont enflammés les murs et le plafond, dégageant de fortes fumées toxiques. Les rapports officiels [GL10] expliquent que la fumée noire recouvrait le sol de l'établissement en 90 secondes et qu'après 3 minutes, l'ensemble du bâtiment était en feu. Les personnes ont tenté de quitter la boite de nuit mais sont quasiment toutes passées par la porte d'entrée au lieu d'utiliser les sorties de secours, créant un engorgement qui a ralenti la fuite. Ces fumées toxiques ainsi que l'incendie, combiné à l'engorgement dans le hall d'entrée, ont fait 100 victimes [TM07].

Contrairement à la tragédie du Kiss Nightclub, l'établissement du Station Nightclub était aux normes anti-incendie en vigueur aux États-Unis au moment du drame. Il y avait une alarme incendie qui s'est déclenchée 20 secondes après le départ du feu, les panneaux lumineux indiquaient bien les sorties qui étaient multiples et la boite de nuit n'était pas en surpopulation excessive. Les décès sont survenus à cause de l'engorgement des personnes qui ont majoritairement évacué par l'entrée principale puis en cassant les fenêtres à coté de l'entrée, comme le montre le tableau 6.5. Peu de personnes sont sorties par la scène car elle a vite pris feu et peu de personnes sont sorties par la cuisine car elle était réservée aux membre du personnel.

TABLEAU 6.5 – Nombre de personnes évacuant le Station Nightclub en fonction de l'emplacement de la sortie la nuit du drame

sortie	principale	bar	cuisines	scène	fenêtres
nombre de personnes	128	78	17	24	105

Le cas du Station Nightclub est intéressant à étudier car il est très documenté, jusque dans les liens sociaux qu'entretenaient les personnes avant l'incident [Bar12]. Ainsi, 72% des personnes présentes lors du drame avaient déjà visitées l'établissement avant l'incendie et 98% des survivants témoignent avoir vu le départ du feu. Seulement 10% des personnes dans le bâtiment étaient venues seules, les autres étaient en groupe de 2 à 5 ou plus, des groupes composés d'amis ou de collègues de travail. Enfin, la fumée était plus nocive dans le Station Nightclub que la moyenne [ETFAB17], ce qui a causé des décès plus rapidement que dans le cas du Kiss Nightclub.

Adaptation du modèle de la boite de nuit brésilienne à la boite de nuit américaine

Le but de ce second exemple étant de montrer que le comportement développé grâce à BEN en section 6.1.2 pour l'évacuation du Kiss Nightclub peut s'adapter à d'autres évacuations de bâtiment, la modélisation de l'incident du Station Nightclub s'est faite en suivant celle du Kiss Nightclub. La boite de nuit a été reconstituée numériquement en suivant ses plans comme le montre la figure 6.5 où les 465 agents simulant les acteurs humains, représentés par des triangles blancs à bord noir, ont été placés aléatoirement pour l'initialisation. Les murs et obstacles qui empêchent le déplacement des agents sont en noir, les sorties sont en bleu et le feu, représenté par la sphère jaune, a été replacé approximativement à l'endroit où il a démarré.

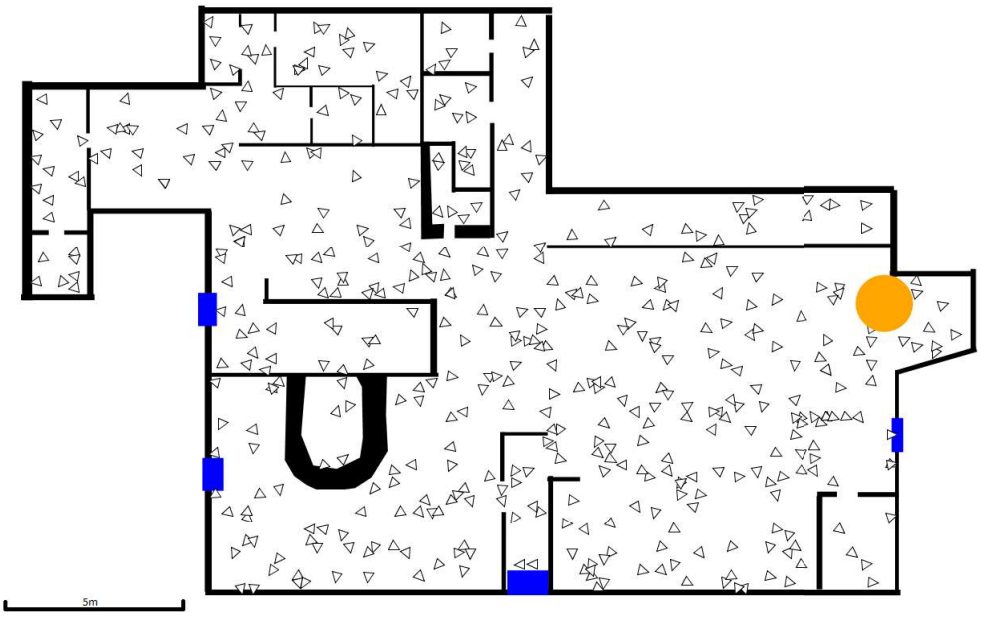

FIGURE 6.5 – Modélisation du Station Nightclub au moment de l'incendie

La propagation de la fumée a été modifiée par rapport à la modélisation du cas du Kiss Nightclub afin de s'approcher d'un établissement quasiment entièrement dans la fumée en 90 secondes. De même, puisque la fumée a été jugée comme très nocive [GL10], tuant les gens en seulement quelques secondes contre près d'une minute dans le cas du Kiss Nightclub, nous avons augmenter l'impact de la fumée sur la santé des personnes.

Le comportement des agents est le même dans le cas du Station Nightclub que dans le cas du Kiss Nightclub. Cela signifie que les agents s'amusent jusqu'à ce qu'ils perçoivent un danger. En fonction de ce qu'ils perçoivent, ils vont évacuer de différentes façons, en suivant les comportements décrits en section 6.1.2. Pour autant, quelques ajustements ont été réalisés afin de se rapprocher du cas réel. Ainsi, les agents perçoivent l'alarme 20 secondes après le début de l'incendie, ajoutant alors l'incertitude qu'il y ait une catastrophe. Cette incertitude permet ainsi la création de peur, qui indiquera à l'agent de fuir si son intensité dépasse un certain seuil. De même, l'obligation de fuir en suivant les panneaux lumineux ne va pas amener les agents vers les toilettes mais vers

la sortie la plus proche. De plus, nous avons repris à l'identique les paramètres discutés en section 6.1.3. Le tableau 6.6 résume les adaptations effectuées sur l'implémentation des plans d'actions et normes décrits en section 6.1.2 pour le comportement d'évacuation des agents dans le cas du Station Nightclub.

TABLEAU 6.6 – Adaptation du comportement d'évacuation pour le Station Nightclub

énoncé du comportement	adaptations
fuite vers l'emplacement précis de la sortie, fuite dans la direction de la sortie, fuite à travers la fumée vers l'emplacement précis de la sortie	Pour le choix de la sortie, les agents sélectionnent la sortie ouverte la plus proche d'eux, c'est à dire celle du bas et du coin inférieur gauche sur l'image 6.5. La sortie à droite était bloquée par les agents de sécurité, la réservant pour les artistes, et devient ouverte pour tous après 50 secondes; La sortie en haut à gauche était réservée au personnel, elle n'est donc pas été ouverte au public.
fuite en respectant la loi indiquant aux agents de suivre les panneaux de sortie de secours	les agents se dirigent vers la sortie la plus proche d'eux.
échapper à la fumée	aucuns changements
suivre les autres agents en fonction des niveaux de confiance	aucuns changements
aide envers un ami	aucuns changements
déplacement aléatoire dans la fumée si l'agent est complètement perdu	aucuns changements

Pour modéliser la congestion, le déplacement des agents est modifié si ceux-ci sont entourés de personnes dans un rayon de 1 mètre. La vitesse est alors proportionnelle au nombre de personnes dans ce cercle de rayon 1 mètre autour de l'agent. En se dirigeant vers la porte de sortie principale, les agents sont obligés de passer par un couloir ce qui va augmenter leur concentration et donc réduire fortement leur vitesse. La congestion n'avait pas été prise en compte dans le cas du Kiss Nightclub car les rapports montraient qu'elle ne faisait pas partie des facteurs ayants causés des décès, ce qui n'est pas le cas du Station Nightclub.

Le modèle complet se trouve au lien suivant : https ://github.com/mathieuBourgais/ExempleThese

6.2.2 Résultats et commentaires

Résultats des simulations du Station Nightclub

Comme pour le cas du Kiss Nightclub, 10 simulations ont été effectuées pour l'évacuation du Station Nightclub afin d'obtenir une moyenne d'agents décédés par simulation. Le résultat graphique d'une de ces simulations, via la plateforme GAMA, est donné par la figure 6.6 où les carrés en teintes de gris représente la quantité de fumée par unité de volume (en blanc, il n'y a aucune fumée, en noir, la zone est entièrement remplie de fumée) et où les agents sont représentés par des triangles de couleur indiquant leur comportement. Les couleurs sont les mêmes que celles évoquées en section 6.1.3, c'est à dire que les agents verts vont vers la sortie parce qu'ils connaissent son emplacement précis, les agents jaunes se dirigent en direction de la porte de sortie, les agents bleus suivent les panneaux lumineux, les agents violets traversent la fumée car ils savent où se trouvent précisément la sortie et les agents blancs ne fuient pas encore. Sur cette image, l'ensemble des comportements possibles pour cette simulation ne sont pas visibles. Une vidéo de la simulation de l'évacuation du Station Nightclub est disponible à l'adresse : https ://github.com/mathieuBourgais/ExempleThese.

FIGURE 6.6 – Résultat graphique de la simulation de l'évacuation du Station Nightclub

Le tableau 6.7 résume les résultats obtenus pour la modélisation de l'évacuation du Station Nightclub en ayant adapté le comportement défini pour l'évacuation du Kiss Nightclub. Ainsi, sur les 465 agents initiaux dans la boite de nuit, il y a en moyenne 98,4 agents morts à la fin de la simulation avec un écart type de 17,0.

TABLEAU 6.7 – Nombre d'agents morts lors de la simulation de la tragédie du Station Nighclub

nombre d'agents	465
moyenne	98,4
écart type	17,0

Lors de l'incident réel du Station Nightclub, il y a eu 100 morts. Le résultat apporté par la simulation est en moyenne proche du résultat réel concernant le nombre d'agents décédés. De plus, la vidéo de la simulation montre de nouveaux motifs comportementaux qui correspondent au cas d'étude. En particulier, la majorité des agents se dirige vers la sortie principale, engorgeant le couloir devant celle-ci ce qui amène aux décès.

Ces résultats montrent que le comportement développé grâce à BEN sur un cas d'évacuation de bâtiment n'est pas spécifique à un cas d'étude et peut être réutilisé sur un autre cas similaire. Cette réutilisation est facilitée par la description de haut niveau du comportement des agents, imposée par l'architecture BEN, qui peut ensuite être spécifié lors de l'implémentation sur des cas précis.

Le tableau 6.8 présente les temps de calcul de chaque pas de temps, en moyenne sur 10 simulations, dans le cas d'étude de l'évacuation du Station Nightclub. Cela signifie qu'il faut en moyenne 711,8 millisecondes pour simuler 1 seconde avec 465 agents initiaux, avec un écart type de 55,9. Comme attendu, ce temps d'exécution est plus faible que pour les simulations du Kiss Nightclub, détaillé dans le tableau 6.4, car le nombre initial d'agents est plus faible.

L'ensemble des résultats apportés par la simulation de l'évacuation du Station Nightclub confirment les conclusions qui avaient été apportées sur l'architecture BEN par la simulation de l'évacuation du Kiss Nighclub : BEN permet de définir des comportements complexes, profitant d'une description de haut niveau et apportant des explications détaillées en termes d'éléments cognitifs, affectifs et sociaux, sur les actions d'agents simulant des acteurs humains, tout en offrant des résultats

TABLEAU 6.8 – Temps de calcul, en millisecondes, obtenus pour la simulation de la tragédie du Station Nightclub

nombre d'agent	465
moyenne	711,8
écart type	55,9

crédibles par rapport à un cas réel. De plus, l'architecture permet de conserver des temps de calcul raisonnables sur plusieurs centaines d'agents, atteignant son objectif de passage à l'échelle.

Commentaires sur la modélisation du Station Nightclub

Le but de cet exemple était de montrer que le comportement développé dans un cas de figure avec l'architecture BEN pouvait être transposé, avec un minimum de transformations, à un cas similaire, apportant là aussi des résultats proches de la réalité. Pour autant, les études autour du Station Nighclub [GL10] [TM07] [Bar12] indiquent que la modélisation de l'évacuation de la boite de nuit peut être différente de celle proposée en section 6.2.1.

La propagation du feu n'a pas été prise en compte dans l'exemple développé dans cette section alors que les rapports font état d'un incendie violent et rapide, où l'ensemble de la boite de nuit s'est retrouvée enflammée en 3 minutes. Il est précisé dans les rapports que la très grande majorité des survivants sont sortis dans les 150 premières secondes. Nous avons donc préféré conserver le modèle de propagation de la fumée développé pour l'étude du Kiss Night Club, en adaptant la vitesse et la dangerosité de cette fumée, qui semble avoir été l'élément le plus meurtrier de ce cas d'étude.

Le comportement des personnes ayant réellement vécues l'incendie du Station Nightclub est aussi différent du comportement des gens ayant vécue l'incident du Kiss Nightclub. L'établissement américain est plus petit donc plus rapide à traverser, ce qui signifie que peu de gens se sont perdus. De plus, les systèmes de sécurité fonctionnaient et une grande proportion de gens étaient des habitués du club, ce qui n'était pas le cas du Kiss Nightclub. Là encore, cela indique que les agents n'avaient aucune raison de ne pas aller rapidement vers la sortie du bâtiment. Tous ces éléments sont des ajustements mineurs qui ont pu être pris en compte sans modifier la description de haut niveau du comportement global.

Par contre, les relations d'amitiés multiples n'ont pas été prises en compte car elles impliquaient d'importantes modifications du comportement. C'est cet aspect qu'a exploré El-Tawil [ETFAB17] dans une étude menée sur le Station Nightclub où le comportement des agents est modélisé par la méthode des champs scalaires. Les agents sont représentés par des particules chargées électriquement et l'environnement est muni de différents attracteurs et répulsifs. Les auteurs proposent d'ajouter une volonté propre à ces particules, modélisée par un attracteur dynamique : avant de fuir, les groupes d'amis vont s'attirer pour se réunir puis l'attracteur disparaît, amenant les agents à considérer les sorties de secours comme nouveaux attracteurs. Si cette approche offre des résultats proches de la réalité sur le nombre de personnes sorties vivantes tout en reproduisant un comportement social proche de celui récupéré par les témoignages lors du vrai drame, il n'en reste pas moins un modèle fondé sur des équations mathématiques avec de nombreux paramètres à initialiser. Il y a donc peu de chances pour que le modèle soit généralisable et son pouvoir d'expressivité, que ce soit dans la définition des comportements ou dans les actions observées, est assez faible.

Une autre simulation de l'évacuation du Station Nightclub est proposée par Valette [VG18]. Le principe de ce travail, qui utilise l'architecture BEN, était de reproduire l'événement réel, le comportement des agents a donc été centré autour de leurs relations sociales. Là aussi, les agents essayent de se rassembler par groupe d'amis avant d'évacuer, mais l'utilisation de BEN permet, contrairement au travail précédent, de conserver une explication de haut niveau sur le comportement des agents. La congestion est aussi prise en compte, tout comme l'alarme ou même les

employés de la boite de nuit essayant de guider les gens vers la sortie. Pour autant, les simulations qui en sont tirées n'arrivent pas à s'approcher du nombre de morts du cas réel, les agents restant trop longtemps dans la boite de nuit à chercher leurs amis avant d'évacuer. Ce résultat peut aussi amener à questionner, au moins partiellement, les témoignages récupérés à l'issue de l'incident où les gens ont expliqué avoir fui entre amis. Dans la précipitation, il n'est pas improbable que chacun ait d'abord cherché à se sauver le plus rapidement possible plutôt que d'attendre plusieurs secondes pour s'assurer que tout le groupe d'amis était prêt à évacuer.

Ces autres approches montrent qu'il est possible de modéliser l'évacuation du Station Nightclub différemment de la façon présentée en section 6.2.1. Ces autres modélisations répondent à d'autres buts et apportent des résultats différents à analyser. Pour autant, cet exemple montre que BEN permet de modéliser le comportement humain avec une description générique tout en offrant des résultats proches d'un cas réel, sans sacrifier les capacités d'explication des actions des agents.

6.3 Conclusion

Ce chapitre présente deux exemples d'application de l'architecture BEN, présentée dans le chapitre 4, sur des cas d'évacuation de bâtiment. Les reproductions de l'évacuation du Kiss Nightclub et du Station Nightclub montrent que l'architecture est utilisable à l'heure actuelle, via son implémentation dans la plateforme de simulation GAMA présentée dans le chapitre 5.

Ces deux exemples montrent que BEN atteint ses objectifs en permettant de définir des comportements complexes et réalistes pour des agents simulant des acteurs humains. Ces comportements sont fondés sur des concepts de haut niveau faisant intervenir des dimensions cognitives, affectives et sociales, permettant de garder un haut degré d'explication dans les actions observées. De plus, ces comportements ne sont pas spécifiques à un cas d'étude et peuvent être adaptés à un ensemble de cas similaires, en conservant une description globale similaire. Enfin, la paramétrisation de tels modèles est simplifiée en étant ramenée aux cinq paramètres du modèle OCEAN de la personnalité. Ainsi, l'architecture offre des comportements complexes et réalistes tout en facilitant son utilisation et sa paramétrisation pour des simulations sur plusieurs centaines d'agents avec un temps d'exécution raisonnable.

Enfin, ces deux études de cas réels montrent que BEN permet de modéliser un comportement d'acteurs humains de façon crédible avec un minimum de concepts à manipuler. Le comportement des agents évacuant les bâtiments se décompose en une suite de processus de haut niveaux qui reposent sur des concepts cognitifs, affectifs et sociaux généraux. Ce modèle descriptif de l'évacuation de bâtiment peut ensuite être spécialisé lors de l'implémentation sur un cas particulier. Cette implémentation et elle aussi facilité par la traduction directe de chaque processus de l'architecture en un *statement* de la plateforme GAMA. De plus, la modularité de BEN permet de réaliser des modèles et des simulations incrémentales, ajoutant des éléments de comportement au fur et à mesure en contrôlant chaque nouvel ajout avec précision. Finalement, la somme de comportements simples permet de créer des simulations riches avec des résultats crédibles par rapport aux événements étudiés.

Conclusion

L'étude d'acteurs humains à travers les simulations sociales s'est étendue ces dernières années à travers les sciences humaines et sociales. Si les premiers modèles proposaient de représenter les acteurs humains par des agents simples, au comportement réactif ou mimant des processus physique comme l'écoulement des fluides ou les déplacements de particules dans des champs de force, ces modélisations, simple à comprendre et à maîtriser, ont montrées leurs limites dans la création de comportements crédibles.

Une approche convaincante pour répondre à ce défi est de s'appuyer sur les mécanismes de raisonnement et de prise de décision humains. Les sciences cognitives révèlent que le prise de décision est un phénomène complexe, pas uniquement rationnel, qui implique la prise en compte d'informations affectives et sociales. Plusieurs de ces dimensions ont été identifiées et des études ont été menées afin de comprendre leur fonctionnement. Le défi, du point de vue de la simulation sociale, consiste à reproduire ces phénomènes lors de la prise de décision des agents sociaux.

Pour répondre à ce défi, nous avons proposé dans cette thèse l'architecture comportementale BEN qui prend en compte une cognition fondée sur le modèle BDI à laquelle viennent s'ajouter les émotions, la personnalité, la contagion émotionnelle, les relations sociales et la gestion des normes. L'ensemble de ces notions s'exprime sous la forme de bases de connaissances autour desquelles s'articulent des modules composés de processus visant à adapter l'action de l'agent à son contexte.

Ce travail de thèse a ainsi cherché à lever les 4 problématiques scientifiques présentés en introduction :

Problématique 1 : Formaliser les notions relatives à la cognition, aux émotions, à la personnalité, à la contagion émotionnelle, aux relations sociales et à la gestion des normes dans un même cadre.

Réponse 1 : Le chapitre 3 propose une formalisation des concepts liés à la cognition, aux émotions, à la personnalité, à la contagion émotionnelle, aux relations sociales et à la gestion des normes. La définition de ces concepts repose sur une représentation générique de l'environnement, basée sur des prédicats. Ainsi, l'agent possède un ensemble de connaissances qui bien que de types différents repose sur une même représentation du monde.

Problématique 2 : Intégrer toutes ces notions cognitives, affectives et sociales dans une architecture comportementale pour modéliser la prise de décisions d'agents simulant des acteurs humains.

Réponse 2 : Le chapitre 4 explique le fonctionnement de l'architecture BEN qui inclut les notions cognitives, affectives et sociales pour la définition du comportement d'agents sociaux. Plus précisément, l'architecture repose sur un moteur cognitif fondé sur le modèle BDI. Autour de ce moteur cognitif, différents modules s'agencent pour offrir à l'agent des moyens de modifier ses connaissances en fonction de perceptions cognitives, affectives et sociales de son environnement. La personnalité participe ensuite à lier les différentes dimensions et les multiples processus, qui sont inter-connectés mais non inter-dépendants.

Problématique 3 : Rendre cette architecture la plus générique possible afin de faciliter son utilisation dans différents domaines.

Réponse 3 : L'architecture BEN a été créée de façon générique et propose un comportement pour les agents qui n'est relié à aucun domaine d'application spécifique. Cette proposition repose sur le formalisme développé dans le chapitre 3. Celui-ci prend pour brique élémentaire le prédicat

qui peut représenter tous les élément de l'environnement, des faits aux actions et qui est donc très versatile.

De plus, comme cela a été montré dans le chapitre 4, l'architecture BEN est modulaire. Tous les éléments ne sont pas obligatoires et peuvent être utilisés comme le souhaite le modélisateur. Celui-ci peut n'utiliser que les modules et processus jugés nécessaires à son cas d'étude sans parasiter le calcul du comportements des agents par des notions qui n'auraient pas de sens par rapport au système étudié.

Enfin, l'architecture propose des moteurs cognitifs, normatifs, émotionnels et sociaux dont le fonctionnement est automatique du point de vue du modélisateur. Pour autant, l'ensemble des paramètres internes à ces moteurs se ramène aux cinq dimensions du modèle OCEAN de la personnalité, facilitant la paramétrisation du comportement des agents utilisant ces moteurs.

Problématique 4 : Proposer une implémentation de cette architecture qui soit le plus simple possible vis à vis d'un public non informaticien et pouvant être utilisée pour des simulations de plusieurs centaines d'agents avec un temps d'exécution raisonnable.

Réponse 4 : Pour faciliter son utilisation, l'architecture BEN a été implémentée dans la plateforme GAMA. D'une part, GAMA est une plateforme populaire dans la communauté des simulations sociales et qui est connue pour sa facilité d'utilisation. L'implémentation de l'architecture, présentée dans le chapitre 5, repose donc sur les principes du langage de programmation de la plateforme pour conserver cette facilité de prise en main. Chaque processus de l'architecture est traduit à l'aide d'un *statement*, facilitant l'implémentation d'un modèle défini avec BEN.

L'implémentation qui est proposée est faite pour s'adapter à l'utilisateur et le contraindre au minimum. Ainsi, un grand nombre *d'opérateurs* et *d'actions* ont été définis et l'architecture possède de nombreuses options. Le modélisateur peut ainsi accéder à l'ensemble des connaissances de l'agent et travailler comme il le souhaite, soit avec une description détaillée et en utilisant de nombreux processus, soit avec des prédicats et des connaissances de plus haut niveau.

Enfin, le chapitre 6 montre que BEN peut être utilisé pour modéliser une situation d'évacuation et que son implémentation offre des temps de calcul raisonnables sur plusieurs centaines d'agent (885 ms en moyenne pour simuler 1 seconde avec 1400 agents initialement dans la boite de nuit à évacuer). Ces deux exemples montrent aussi que BEN permet de définir un comportement crédible tout en gardant une description de haut niveau. De plus, ce comportement est assez générique pour être adapté sur un second cas d'évacuation tout en conservant des résultats crédibles.

Perspectives

Le chapitre 6 montre que BEN est utilisable à l'heure actuelle et qu'il remplit ses objectifs dans la définition de comportements d'agents sociaux pour un résultat crédible. Aussi, BEN a été utilisé sur d'autres cas d'étude par d'autres modélisateurs : pour étudier le comportement d'agriculteurs au Vietnam [TTG+15], pour étudier l'utilisation de bâtiments à faible consommation d'énergie [TMT17] ou encore pour étudier l'évacuation d'une population lors des incendies de brousse en Australie [ATDG17]. Ces différentes utilisations montrent que BEN offre des comportements plus crédibles que d'autres architectures comportementales [Tru16] et est jugé plus simple à utiliser qu'une machine à états finis[ATDG17].

Pour autant, il serait intéressant de tester l'architecture et son implémentation sur d'autres cas d'étude, qu'il s'agisse d'évacuation ou d'autres domaines d'études. Le but serait de montrer que l'architecture peut s'adapter à n'importe quel contexte grâce à son formalisme basé sur le concept généraliste de prédicat pour représenter l'environnement des agents. L'utilisation de l'architecture par un public plus large permettrait aussi de récupérer des retours sur la facilité d'utilisation ainsi que sur les possibilités de comportement offertes pour des situations précises.

À l'avenir, BEN pourrait être amené à évoluer. La version présentée dans cette thèse représente une base qu'il est possible de compléter. Ainsi, il pourrait être ajouté un processus permettant de gérer l'impact de la création des émotions sur les connaissances de l'agent, qui interviendrait entre

le moteur émotionnel et le moteur cognitif et qui ferait référence à la notion de *coping* du modèle émotionnel de Smith et Lazarus [SL+90]. Sur le même modèle, il pourrait être ajouté un système d'oubli sur les relations sociales, indiquant que si une relation n'est pas mise à jour depuis un certain temps, elle disparaîtrait des bases de connaissance de l'agent.

Toujours sur les notions déjà exploitée par BEN, il serait intéressant d'offrir encore plus de flexibilité aux utilisateurs en leur permettant de redéfinir, s'ils le souhaitent, les différents processus automatiques comme le moteur émotionnel, le moteur social, le moteur normatif ou le moteur cognitif. Ainsi, un utilisateur expert aurait la possibilité de proposer des moteurs de prise de décision personnels ou encore de tester d'autres théories émotionnelles que la théorie OCC directement incluse dans l'architecture. Ainsi, il serait possible de tester différentes théories issues des sciences cognitives, permettant de choisir celle qui s'applique le mieux au cas étudié.

Enfin, d'autres dimensions cognitives, affectives ou sociales pourraient être incorporées dans l'architecture, tout en conservant sa modularité et son découpage en quatre étapes principales : perception, modification des connaissances, prise de décision et évolution temporelle des connaissances. Les agents pourraient être dotés de mémoires plus complexes et des notions comme la culture, l'apprentissage, la communication ou encore l'expérience, qui viendraient enrichir leur comportement. Il faudrait alors formaliser ces notions, identifier les processus qu'elles regroupent et les connecter avec le reste de l'architecture de sorte que l'ensemble des processus soient interconnectés lors de la prise de décision, mais pas inter-dépendants de sorte à ce qu'ils puissent tous être déconnectés si le modélisateur le souhaite.

Bibliographie

[ACCP07] Giulia Andrighetto, Marco Campennì, Rosaria Conte, and Marco Paolucci. On the immergence of norms : a normative agent architecture. In *Proceedings of AAAI symposium, social and organizational aspects of intelligence*, page 19, 2007. ix, 17, 18, 23

[ACTP07] Giulia Andrighetto, Rosaria Conte, Paolo Turrini, and Mario Paolucci. Emergence in the loop : Simulating the two way dynamics of norm innovation. In *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2007. 23

[Ada07] Carole Adam. *Emotions : from psychological theories to logical formalization and implementation in a BDI agent.* Phd thesis, 2007. 33, 43, 50, 65, 66

[AG16] Carole Adam and Benoit Gaudou. Bdi agents in social simulations : a survey. *The Knowledge Engineering Review*, 2016. 3, 30, 42

[AP06] Frédéric Amblard and Denis Phan. *Modélisation et simulation multi-agents : Applications pour les Sciences de l'Homme et de la Société.* Hermes science publ., 2006. 7

[ARB+06] Frédéric Amblard, Juliette Rouchier, Pierre Bommel, Franck Varenne, and Denis Phan. *Evaluation et validation de modèles multi-agents.* Hermes Science Publications, 2006. 111

[Arn60] Magda B Arnold. *Emotion and personality.* Columbia University Press, 1960. 31

[ATDG17] Carole Adam, Patrick Taillandier, Julie Dugdale, and Benoit Gaudou. Bdi vs fsm agents in social simulations for raising awareness in disasters : A case study in melbourne bushfires. *International Journal of Information Systems for Crisis Response and Management (IJISCRAM)*, 9(1) :27–44, 2017. 97, 120

[Ati13] B Atiyeh. Brazilian kiss nightclub disaster. *Annals of burns and fire disasters*, 26(1) :3, 2013. 100, 101, 109, 112

[Axe97] Robert Axelrod. Advancing the art of simulation in the social sciences. In *Simulating social phenomena*, pages 21–40. Springer, 1997. 2, 13

[BA98] MD Byrne and JR Anderson. Perception and action. *The atomic components of thought*, 1998. 16, 23, 28, 42

[Bar02] Sigal G Barsade. The ripple effect : Emotional contagion and its influence on group behavior. *Administrative Science Quarterly*, 47(4) :644–675, 2002. 36

[Bar12] John Barylick. *Killer Show : The Station Nightclub Fire, America's Deadliest Rock Concert.* UPNE, 2012. 114, 117

[BBC+15] Laurent Bègue, Jean-Léon Beauvois, Didier Courbet, Dominique Oberlé, Johan Lepage, and Aaron A Duke. Personality predicts obedience in a milgram paradigm. *Journal of Personality*, 83(3) :299–306, 2015. 34, 61, 74

[BBCP05] Fabio Bellifemine, Federico Bergenti, Giovanni Caire, and Agostino Poggi. Jade—a java agent development framework. In *Multi-Agent Programming*, pages 125–147. Springer, 2005. 8

[BBLPB16] Nicolas Becu, Pierre Bommel, Christophe Le Page, and François Bousquet. Cormas, une plate-forme multi-agent pour concevoir collectivement des modèles et interagir avec les simulations. In *Journées Francophones sur les Systèmes Multi-Agents (JFSMA)*. Cépaduès, 2016. 2, 11

[BBPLP98] François Bousquet, Innocent Bakam, Hubert Proton, and Christophe Le Page. Cormas : common-pool resources and multi-agent systems. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 826–837. Springer, 1998. 11

[BC01] Timothy Bickmore and Justine Cassell. Relational agents : a model and implementation of building user trust. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 2001. 38

[BDH+01] Jan Broersen, Mehdi Dastani, Joris Hulstijn, Zisheng Huang, and Leendert van der Torre. The boid architecture : conflicts between beliefs, obligations, intentions and desires. In *Proceedings of the fifth international conference on Autonomous agents*, pages 9–16. ACM, 2001. 18, 23, 44, 52, 71

[BDM+09] Tibor Bosse, Rob Duell, Zulfiqar Ali Memon, Jan Treur, and C Natalie Van Der Wal. Multi-agent model for mutual absorption of emotions. *ECMS*, 2009 :212–218, 2009. 22, 37, 44, 52, 62

[BG14] Tina Balke and Nigel Gilbert. How do agents make decisions? a survey. *Journal of Artificial Societies and Social Simulation*, 17(4) :13, 2014. ix, 18, 19, 26

[BL87] Penelope Brown and Stephen C Levinson. *Politeness : Some universals in language usage*, volume 4. Cambridge university press, 1987. 38

[BL06] Bradley J Best and Christian Lebiere. Cognitive agents interacting in real and virtual worlds. *Cognition and multi-agent interaction : From cognitive modeling to social simulation*, pages 186–218, 2006. 17, 23

[BLPM02] François Bousquet, Christophe Le Page, and Jean-Pierre Müller. Modélisation et simulation multi-agent. *Assises nationales du GDR I3*, 2002. 1

[Bra87] M Bratman. *Intentions, plans, and practical reason*. Harvard Univ. Press, 1987. 2, 13, 23, 28, 49, 69

[Bra15] Johannes Brauer. The visualworks development environment. In *Programming smalltalk–Object-orientation from the beginning*, pages 77–96. Springer, 2015. 11

[Bre02] Laurent Breton. *GranuLab : Un système d'aide à la découverte scientifique pour la physique des milieux granulaires*. PhD thesis, Paris 6, 2002. 1

[BRM+09] Michael Balmer, Marcel Rieser, Konrad Meister, David Charypar, Nicolas Lefebvre, Kai Nagel, and K Axhausen. Matsim-t : Architecture and simulation times. *Multi-agent systems for traffic and transportation engineering*, 2009. 16

[Bru17] Jerome Bruner. *A study of thinking*. Routledge, 2017. 25, 27

[BTVA18] Mathieu Bourgais, Patrick Taillandier, Laurent Vercouter, and Carole Adam. Emotion modeling in social simulation : A survey. *Journal of Artificial Societies and Social Simulation*, 21(2) :5, 2018. 3, 19, 31

[BVDTV06] Guido Boella, Leendert Van Der Torre, and Harko Verhagen. Introduction to normative multiagent systems. *Computational & Mathematical Organization Theory*, 12(2-3) :71–79, 2006. 17

[Can27] Walter B. Cannon. The james-lange theory of emotions : A critical examination and an alternative theory. *The American Journal of Psychology*, 39(1/4) :106–124, 1927. 31

[Cat66] Raymond B Cattell. The scree test for the number of factors. *Multivariate behavioral research*, 1(2) :245–276, 1966. 34

[CC05] C. Chivas and J. Cescon. Formalisation du savoir et des outils dans le domaine des risques majeurs (dra-35) - toxicité et dispersion des fumées d'incendie phénoménologie et modélisation des effets. Technical report, INERIS, 2005. 102

[CC+16] Rosaria Conte, Cristiano Castelfranchi, et al. *Cognitive and social action*. Garland Science, 2016. 22

[CDJT99] Cristiano Castelfranchi, Frank Dignum, Catholijn M Jonker, and Jan Treur. Deliberative normative agents : Principles and architecture. In *International Workshop on Agent Theories, Architectures, and Languages*, pages 364–378. Springer, 1999. 22

[CHSN97] Patrick Coquillard, David RC Hill, and T Sime-Ngando. *Modélisation et simulation d'écosystèmes : des modèles déterministes aux simulations à évènements discrets*. Masson Paris, 1997. 9

[CL90] Philip R Cohen and Hector J Levesque. Intention is choice with commitment. *Artificial intelligence*, 42(2-3) :213–261, 1990. 29, 44

[CM94] Nancy L Collins and Lynn Carol Miller. Self-disclosure and liking : a meta-analytic review. *Psychological bulletin*, 1994. 69

[CM10] Ibrahim Cil and Murat Mala. A multi-agent architecture for modelling and simulation of small military unit combat in asymmetric warfare. *Expert Systems with Applications*, 37(2) :1331–1343, 2010. 2

[Col03] Nick Collier. Repast : An extensible framework for agent simulation. *The University of Chicago's Social Science Research*, 36 :2003, 2003. 2, 10

[CRPL+04] Claudio Cioffi-Revilla, Sean Paus, Sean Luke, James L Olds, and Jason Thomas. Mnemonic structure and sociality : a computational agent-based simulation model. In *Proceedings of the Conference on Collective Intentionality IV*, pages 1–11, 2004. 10

[Dam94] Antonio Damasio. *Descartes' Error, Emotion Reason and the Human Brain*. Grosset/-Putnam, 1994. 2, 19, 25, 30, 36, 42

[Dav02] Paul Davidsson. Agent based social simulation : A computer science view. *Journal of artificial societies and social simulation*, 5(1), 2002. 9

[DDJ08] Frank Dignum, Virginia Dignum, and Catholijn M Jonker. Towards agents for policy making. In *International Workshop on Multi-Agent Systems and Agent-Based Simulation*, pages 141–153. Springer, 2008. ix, 18, 19, 23

[Dig99] Frank Dignum. Autonomous agents with norms. *Artificial intelligence and law*, 7(1) :69–79, 1999. 17, 41

[dRG86] Joseph de Rivera and Carmen Grinkis. Emotions as social relationships. *Motivation and emotion*, 10(4) :351–369, 1986. 38, 68

[E+64] Hans Jurgen Eysenck et al. *Manual of the Eysenck personality inventory*. University of London Press London, 1964. 34

[EE+87] Hans J Eysenck, Michael W Eysenck, et al. *Personality and individual differences*. Plenum New York, NY, 1987. 34

[ETFAB17] Sherif El-Tawil, Jieshi Fang, Benigno Aguirre, and Eric Best. A computational study of the station nightclub fire accounting for social relationships. *Journal of Artificial Societies and Social Simulation*, 20(4) :10, 2017. 114, 117

[Eys50] Hans J Eysenck. *Dimensions of personality*, volume 5. Transaction Publishers, 1950. 34

[Eys90] Hans J Eysenck. Biological dimensions of personality. *Handbook of Personality : Theory and Research*, pages 244–276, 1990. 34

[Eys91] Hans Jurgen Eysenck. Dimensions of personality : 16, 5 or 3 ?—criteria for a taxonomic paradigm. *Personality and individual differences*, 12(8) :773–790, 1991. 34, 35, 50

[FBBB56] Sigmund Freud, Josef Breuer, Anne Berman, and Josef Breuer. *Études sur l'hystérie*. Presses universitaires de France, 1956. 25, 26

[Fer97] Jacques Ferber. *Les systèmes multi-agents : vers une intelligence collective*. InterEditions, 1997. 8

[FKTS89] Nico H Frijda, Peter Kuipers, and Elisabeth Ter Schure. Relations among emotion, appraisal, and emotional action readiness. *Journal of personality and social psychology*, 57(2) :212, 1989. 31

[For92] Joseph Paul Forgas. Affect in social judgments and decisions : A multiprocess model. In *Advances in experimental social psychology*, volume 25, pages 227–275. Elsevier, 1992. 21

[Gar93] Howard Gardner. *Histoire de la révolution cognitive*. Payot, 1993. 2, 27, 42

[Geb05] Patrick Gebhard. Alma : a layered model of affect. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 29–36. ACM, 2005. 21, 23

[Gib65] Jack P Gibbs. Norms : The problem of definition and classification. *American Journal of Sociology*, 70(5) :586–594, 1965. 39

[GL10] K. GILL and E. A. Laposata. Factors impacting injury and fatality in the station nightclub fire. Technical report, 2010. 2, 113, 114, 117

[GM04] Jonathan Gratch and Stacy Marsella. A domain-independent framework for modeling emotion. *Cognitive Systems Research*, 2004. ix, 20, 23, 33, 43

[GM05] Jonathan Gratch and Stacy Marsella. Evaluating a computational model of emotion. *Autonomous Agents and Multi-Agent Systems*, 11(1) :23–43, 2005. 23

[Gra02] Jonathan Gratch. Socially situated planning. In *Socially Intelligent Agents*, pages 181–188. Springer, 2002. 37

[GT05] Nigel Gilbert and Klaus Troitzsch. *Simulation for the social scientist*. McGraw-Hill Education (UK), 2005. 2, 9

[GTG+13] Arnaud Grignard, Patrick Taillandier, Benoit Gaudou, Duc An Vo, Nghi Quang Huynh, and Alexis Drogoul. *GAMA 1.6 : Advancing the Art of Complex Agent-Based Modeling and Simulation*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. 2, 12, 79, 80

[HCR93] Elaine Hatfield, John T Cacioppo, and Richard L Rapson. Emotional contagion. *Current directions in psychological science*, 2(3) :96–100, 1993. 36, 52

[HJC03] Amy E Henninger, Randolph M Jones, and Eric Chown. Behaviors that emerge from emotion and cognition : implementation and evaluation of a symbolic-connectionist architecture. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 321–328. ACM, 2003. 20

[Hom06] Cars H Hommes. Interacting agents in finance. *Tinbergen Institute Discussion Paper*, (06-029/1), 2006. 1

[HRHL01] Nick Howden, Ralph Rönnquist, Andrew Hodgson, and Andrew Lucas. Jack intelligent agents-summary of an agent infrastructure. In *5th International conference on autonomous agents*, 2001. 8

[HSB02] Jomi Fred Hübner, Jaime Simao Sichman, and Olivier Boissier. A model for the structural, functional, and deontic specification of organizations in multiagent systems. In *Brazilian Symposium on Artificial Intelligence*, pages 118–128. Springer, 2002. 17, 41

[Isb06] Katherine Isbister. *Better game characters by design : A psychological approach*. Elsevier San Francisco, 2006. 38

[Jag17] Wander Jager. Enhancing the realism of simulation (eros) : On implementing and developing psychological theory in social simulation. *Journal of Artificial Societies and Social Simulation*, 20(3) :14, 2017. 2, 15

[JSL09] Hazaël Jones, Julien Saunier, and Domitile Lourdeaux. Personality, emotions and physiology in a BDI agent architecture : The PEP-> BDI model. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology-Volume 02*, pages 263–266. IEEE Computer Society, 2009. 23

[JTLR93] Randolph M Jones, Milind Tambe, John E Laird, and Paul S Rosenbloom. Intelligent automated agents for flight training simulators. Technical report, UNIVERSITY OF SOUTHERN CALIFORNIA MARINA DEL REY INFORMATION SCIENCES INST, 1993. 17, 23, 28, 31

[JVH07] Hong Jiang, Jose M Vidal, and Michael N Huhns. EBDI : an architecture for emotional agents. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, page 11. ACM, 2007. ix, 20, 23

[KB15] Kalliopi Kravari and Nick Bassiliades. A survey of agent platforms. *Journal of Artificial Societies and Social Simulation*, 18(1) :11, 2015. 8

[KBSP14] William Kaye-Blake, Chris Schilling, and Elizabeth Post. Validation of an agricultural mas for southland, new zealand. *Journal of Artificial Societies and Social Simulation*, 17(4) :5, 2014. 2

[KGH14] Adam DI Kramer, Jamie E Guillory, and Jeffrey T Hancock. Experimental evidence of massive-scale emotional contagion through social networks. *Proceedings of the National Academy of Sciences*, 111(24) :8788–8790, 2014. 36

[KH01] Dacher Keltner and Jonathan Haidt. *Social functions of emotions*. Guilford Press, 2001. 68

[Knu96] Brian Knutson. Facial expressions of emotion influence interpersonal trait inferences. *Journal of Nonverbal Behavior*, 1996. 68

[Koh66] Heinz Kohut. Forms and transformations of narcissism. *Journal of the American Psychoanalytic association*, 14(2) :243–272, 1966. 37

[Kol05] Martin Josef Kollingbaum. *Norm-governed practical reasoning agents*. PhD thesis, University of Aberdeen Aberdeen, 2005. ix, 18, 19, 23

[KPWvL01] Jan-Ulrich Kreft, Cristian Picioreanu, Julian WT Wimpenny, and Mark CM van Loosdrecht. Individual-based modelling of biofilms. *Microbiology*, 147(11) :2897–2912, 2001. 1

[Lai12] John E Laird. *The Soar cognitive architecture*. MIT press, 2012. ix, 28

[LCRP+05] Sean Luke, Claudio Cioffi-Revilla, Liviu Panait, Keith Sullivan, and Gabriel Balan. Mason : A multiagent simulation environment. *SIMULATION*, 81(7) :517–527, 2005. 2, 10

[LJ22] Carl Georg Ed Lange and William Ed James. *The emotions, Vol. 1*. Williams & Wilkins Co, 1922. 31

[LLB11] Margaux Lhommet, Domitile Lourdeaux, and Jean-Paul Barthès. Never alone in the crowd : A microscopic crowd model based on emotional contagion. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2011 IEEE/WIC/ACM International Conference on*, volume 2, pages 89–92. IEEE, 2011. 3, 22

[LLB12] Margaux Lhommet, Domitile Lourdeaux, and Jean-Paul A Barthès. Foule sentimentale : influence des caractéristiques individuelles sur la contagion émotionnelle. *Revue d'Intelligence Artificielle*, 26(3) :281–308, 2012. 35, 43

[LNR87] John E Laird, Allen Newell, and Paul S Rosenbloom. Soar : An architecture for general intelligence. *Artificial intelligence*, 1987. 2, 16, 23, 28, 42

[MAS+17] Vinicius Montenegro, Diana Adamatti, Marcos Vinicius Scholl, Bruna Corrêa, and Miguel Zinelli Jr. Multi-agent simulation of a real evacuation scenario : Kiss nightclub and the panic factor. In *EUMAS*, 2017. 100, 111

[MB14] EG Macatulad and AC Blanco. 3dgis-based multi-agent geosimulation and visualization of building evacuation using gama platform. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2014. 97

[MHS10] Nick Malleson, Alison Heppenstall, and Linda See. Crime reduction through simulation : An agent-based model of burglary. *Computers, environment and urban systems*, 34(3) :236–250, 2010. 2

[Mil56] George A Miller. The magical number seven, plus or minus two : Some limits on our capacity for processing information. *Psychological review*, 63(2) :81, 1956. 27

[Mil63] Stanley Milgram. Behavioral study of obedience. *The Journal of abnormal and social psychology*, 67(4) :371, 1963. 61

[MJ92] Robert R McCrae and Oliver P John. An introduction to the five-factor model and its applications. *Journal of personality*, 60(2) :175–215, 1992. 22, 35, 43, 50, 60, 65

[MMM85] Isabel Briggs Myers, Mary H McCaulley, and Robert Most. *Manual, a guide to the development and use of the Myers-Briggs type indicator*. Consulting Psychologists Press, 1985. 34, 43

[MW+93] John-Jules Ch Meyer, Roel J Wieringa, et al. *Deontic logic in computer science normative system specification*. John Wiley and Sons Ltd., 1993. 17

[Mye97] Karen L Myers. User guide for the procedural reasoning system. *SRI International AI Center Technical Report.*, 1997. 13, 23

[NHCV07] Michael J North, Tom R Howe, Nicholson T Collier, and Jerry R Vos. A declarative model assembly infrastructure for verification and validation. In *Advancing social simulation : The first world congress*, pages 129–140. Springer, 2007. 10

[Nor09] Emma Jane Norling. *Modelling human behaviour with BDI agents*. PhD thesis, 2009. 28, 30, 43

[OCC90] Andrew Ortony, Gerald L Clore, and Allan Collins. *The cognitive structure of emotions*. Cambridge university press, 1990. ix, 21, 31, 32, 33, 43, 50, 51, 65

[Occ03] Michel Occello. Méthodologie et architectures pour la conception de systèmes multi-agents. *Mémoire d'Habilitation à Diriger les Recherches*, 2003. 13

[OKC91] Andrew Ortony, William Kessen, and Fergus IM Craik. *Memories, Thoughts and Emotions : Essays in Honour of George Mandler*. LEA, 1991. 38, 68

[Ort02] Andrew Ortony. On making believable emotional agents believable. *Trappl et al.(Eds.)(2002)*, pages 189–211, 2002. 34

[OSC09a] Magalie Ochs, Nicolas Sabouret, and Vincent Corruble. Simulation de la dynamique des émotions et des relations sociales de personnages virtuels. *Revue des Sciences et Technologies de l'Information-Série RIA : Revue d'Intelligence Artificielle*, 23(2-3) :327–357, 2009. ix, 39

[OSC09b] Magalie Ochs, Nicolas Sabouret, and Vincent Corruble. Simulation of the dynamics of nonplayer characters' emotions and social relations in games. *IEEE Transactions on Computational Intelligence and AI in Games*, 1(4) :281–297, 2009. 3, 22, 35, 38, 43, 67

[P+72] Karl Raimund Popper et al. Objective knowledge : An evolutionary approach. *Ox-43*, 1972. 9

[Pav10] P Ivan Pavlov. Conditioned reflexes : an investigation of the physiological activity of the cerebral cortex. *Annals of neurosciences*, 17(3) :136, 2010. 26

[PBL05] Alexander Pokahr, Lars Braubach, and Winfried Lamersdorf. Jadex : A bdi reasoning engine. In *Multi-agent programming*. Springer, 2005. 8

[PF04] Pierre Philippot and Robert S Feldman. Positive emotion and the regulation of interpersonal relationships. In *The regulation of emotion*, pages 142–171. Psychology Press, 2004. 38

[PI01] Helmut Prendinger and Mitsuru Ishizuka. Social role awareness in animated agents. In *Proceedings of the fifth international conference on Autonomous agents*. ACM, 2001. 38

[PL04] Liviu Panait and Sean Luke. A pheromone-based utility model for collaborative foraging. In *Autonomous Agents and Multiagent Systems, 2004. AAMAS 2004. Proceedings of the Third International Joint Conference on*, pages 36–43. IEEE, 2004. 10

[Rao96] Anand S Rao. Agentspeak (l) : Bdi agents speak out in a logical computable language. In *European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, pages 42–55. Springer, 1996. 30

[RCN+16] JA Rincon, A Costa, P Novais, V Julian, and C Carrascosa. A dynamic emotional model for agent societies. In *Advances in Practical Applications of Scalable Multi-agent Systems. The PAAMS Collection*, pages 169–182. Springer, 2016. 21, 31

[RG+95] Anand S Rao, Michael P Georgeff, et al. Bdi agents : from theory to practice. In *ICMAS*, volume 95, pages 312–319, 1995. 25, 29

[RHR98] Daniel Rousseau and Barbara Hayes-Roth. A social-psychological model for synthetic actors. In *Proceedings of the second international conference on Autonomous agents*, pages 165–172. ACM, 1998. 37, 38

[RHSV15] William Rand, Jeffrey Herrmann, Brandon Schein, and Neža Vodopivec. An agent-based model of urgent diffusion in social media. *Journal of Artificial Societies and Social Simulation*, 18(2) :1, 2015. 2

[RM77] James A Russell and Albert Mehrabian. Evidence for a three-factor theory of emotions. *Journal of research in Personality*, 1977. 31, 43

[Rön07] Ralph Rönnquist. The goal oriented teams (gorite) framework. In *International Workshop on Programming Multi-Agent Systems*. Springer, 2007. 16

[RRO15] Nur Raihan Ramli, Sazalinsyah Razali, and Mashanum Osman. An overview of simulation software for non-experts to perform multi-robot experiments. In *ISAMSR*. IEEE, 2015. 97

[RS62] Anne Marie Rocheblave-Spenlé. *La notion de rôle en psychologie sociale : étude historico-critique.* Presses universitaires de France, 1962. 37, 38, 44

[SA04] Gabriel Santos and Benigno E Aguirre. *A critical review of emergency evacuation simulation models.* Disaster Research Center, 2004. 100

[SC09] Bastin Tony Roy Savarimuthu and Stephen Cranefield. A categorization of simulation works on norms. *Dagstuhl Seminar Proceedings*, 2009. 40

[SCKH04] Michelle N Shiota, Belinda Campos, Dacher Keltner, and Matthew J Hertenstein. Positive emotion and the regulation of interpersonal relationships. *The regulation of emotion*, 2004. 68

[Ser00] David Servat. *Modélisation de dynamiques de flux par agents. Application aux processus de ruissellement, infiltration et érosion.* PhD thesis, Paris 6, 2000. 1

[SG10] Frank Schweitzer and David Garcia. An agent-based model of collective emotions in online communities. *The European Physical Journal B*, 2010. 21, 31

[Sim06] John Simpson. Simulations are not models. 2006. 9

[Ski90] Burrhus Frederic Skinner. *The behavior of organisms : An experimental analysis.* BF Skinner Foundation, 1990. 27

[SKS08] Ilias Sakellariou, Petros Kefalas, and Ioanna Stamatopoulou. Enhancing netlogo to simulate bdi communicating agents. In *Hellenic Conference on Artificial Intelligence*. Springer, 2008. 12, 14, 43

[SL+90] Craig A Smith, Richard S Lazarus, et al. Emotion and adaptation. *Handbook of personality : Theory and research*, pages 609–637, 1990. 20, 32, 33, 43, 121

[SMC14] Eliot R Smith, Diane M Mackie, and Heather M Claypool. *Social psychology*. Psychology Press, 2014. 68

[SMP01] Ron Sun, Edward Merrill, and Todd Peterson. From implicit skills to explicit knowledge : A bottom-up model of skill learning. *Cognitive science*, 25(2) :203–244, 2001. 17, 23, 28, 42

[SPL16] Dhirendra Singh, Lin Padgham, and Brian Logan. Integrating bdi agents with agent-based simulation platforms. *Autonomous Agents and Multi-Agent Systems*, 30(6) :1050–1071, 2016. 15

[SS62] Stanley Schachter and Jerome Singer. Cognitive, social, and physiological determinants of emotional state. *Psychological review*, 69(5) :379, 1962. 31

[SSE84] Klaus R Scherer, Klaus R Scherer, and Paul Ekman. On the nature and function of emotion : A component process approach. *Approaches to emotion*, 2293 :317, 1984. 31

[Str02] Tiberiu Stratulat. *Systèmes d'agents normatifs : concepts et outils logiques*. PhD thesis, Université de Caen, 2002. 41

[Sun06] Ron Sun. *Cognition and multi-agent interaction : From cognitive modeling to social simulation*. Cambridge University Press, 2006. ix, 2, 17

[Sun07] Ron Sun. The importance of cognitive architectures : An analysis based on clarion. *Journal of Experimental & Theoretical Artificial Intelligence*, 19(2) :159–193, 2007. 13

[Sve00] Jan Svennevig. *Getting acquainted in conversation : a study of initial interactions*. John Benjamins Publishing, 2000. 22, 37, 44, 55

[TBC+16] Patrick Taillandier, Mathieu Bourgais, Philippe Caillou, Carole Adam, and Benoit Gaudou. A situated BDI agent architecture for the GAMA modelling and simulation platform. In *17th International Workshop on Multi-Agent-Based Simulation (MABS 2016) at AAMAS 2016*, number 10399, pages pp. 3–23, Singapore, Singapore, May 2016. ix, 13, 14, 15, 23, 43, 69, 79

[TBC+17] Patrick Taillandier, Mathieu Bourgais, Philippe Caillou, Carole Adam, and Benoit Gaudou. A bdi agent architecture for the gama modeling and simulation platform. In Luis Gustavo Nardin and Luis Antunes, editors, *Multi-Agent Based Simulation XVII*, pages 3–23, Cham, 2017. Springer International Publishing. 97

[TBDV17] Patrick Taillandier, Mathieu Bourgais, Alexis Drogoul, and Laurent Vercouter. Using parallel computing to improve the scalability of models with bdi agents. In *Social Simulation Conference*, 2017. 98

[TDZ08] Jean-Pierre Treuil, Alexis Drogoul, and Jean-Daniel Zucker. *Modélisation et simulation à base d'agents : exemples commentés, outils informatiques et questions théoriques*. Dunod, 2008. 1, 9

[TFB+11] Jason Tsai, Natalie Fridman, Emma Bowring, Matthew Brown, Shira Epstein, Gal Kaminka, Stacy Marsella, Andrew Ogden, Inbal Rika, Ankur Sheel, et al. Escapes : evacuation simulation with children, authorities, parents, emotions, and social comparison. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 457–464. International Foundation for Autonomous Agents and Multiagent Systems, 2011. 19

[The02] Göran Therborn. Back to norms! on the scope and dynamics of norms and normative action. *Current Sociology*, 50(6) :863–880, 2002. 40

[TM07] Jeffrey Tubbs and Brian Meacham. *Egress design solutions : A guide to evacuation and crowd management planning*. John Wiley & Sons, 2007. 113, 117

[TMT17] Franck Taillandier, Alice Micolier, and Patrick Taillandier. Li-bim. Technical report, 2017. Retrieved from : `https://www.comses.net/codebases/5745/releases/1.0.0/`. 120

[Tru16] Chi Quang Truong. *Integrating cognitive models of human decision-making in agent-based models : an application to land use planning under climate change in the Mekong river delta*. PhD thesis, Université Pierre et Marie Curie-Paris VI, 2016. 120

[TTG+15] Quang Chi Truong, Patrick Taillandier, Benoit Gaudou, Minh Quang Vo, Trung Hieu Nguyen, and Alexis Drogoul. Exploring agent architectures for farmer behavior in land-use change. a case study in coastal area of the vietnamese mekong delta. In *International Workshop on Multi-Agent Systems and Agent-Based Simulation*, pages 146–158. Springer, 2015. 95, 120

[Tuo95] Raimo Tuomela. The importance of us : A philosophical study of basic social notions. 1995. 39, 52

[VDPBB+06] H. Van Dyke Parunak, Robert Bisson, Sven Brueckner, Robert Matthews, and John Sauter. A model of emotions for situated agents. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 993–995. ACM, 2006. ix, 20, 21, 23

[vdWFR+17] C Natalie van der Wal, Daniel Formolo, Mark A Robinson, Michael Minkov, and Tibor Bosse. Simulating crowd evacuation with socio-cultural, cognitive, and emotional elements. In *Transactions on Computational Collective Intelligence XXVII*, pages 139–177. Springer, 2017. 22

[VG18] Marion Valette and Benoit Gaudou. Modeling a real-case situation of egress using bdi agents with emotions and social skills. In *PRIMA*, 2018. 117

[Wat13] John B Watson. Psychology as the behaviorist views it. *Psychological review*, 20(2) :158, 1913. 25, 26, 27

[WC92] David Watson and Lee Anna Clark. On traits and temperament : General and specific factors of emotional experience and their relation to the five-factor model. *Journal of personality*, 60(2) :441–476, 1992. 34

[WCW97] Marilyn A Walker, Janet E Cahn, and Stephen J Whittaker. Improvising linguistic style : Social and affective bases for agent personality. In *Proceedings of the first international conference on Autonomous agents*, pages 96–105. ACM, 1997. 38

[WE99] Uri Wilensky and I Evanston. Netlogo : Center for connected learning and computer-based modeling. *Northwestern Univ., Evanston, IL*, 1999. 2, 11

[Woo09] Michael Wooldridge. *An introduction to multiagent systems*. John Wiley & Sons, 2009. 1, 8

[yLLd06] Fabiola López y López, Michael Luck, and Mark d'Inverno. A normative framework for agent-based systems. *Computational & Mathematical Organization Theory*, 12(2-3) :227–250, 2006. 41, 44, 52

[Zaj65] Robert B Zajonc. Social facilitation. *Science*, 149(3681) :269–274, 1965. 38